 ISSN 2069-7988
ISSN-L 2069-7988

Culegere de probleme

Sisteme de gestiune a bazelor de date distribuite

SGBDD

Aplicatii in Visual Foxpro(9.0) / Oracle(9i)

Nicusor Zlota -Focsani

 ISSN 2069-7988
ISSN-L 2069-7988

CAPITOLUL 1
Generalităţi şi structurile de date Oracle 9i

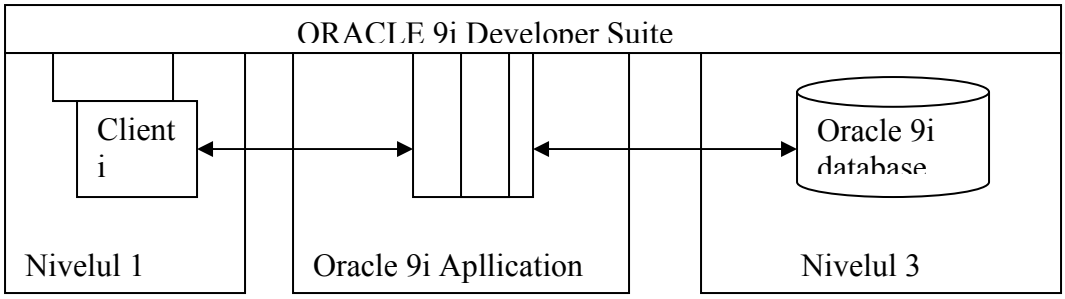
1.2 Generalităţi despre platforma Oracle 9i

Oracle 9i este un sistem de gestiune a bazelor de date relaţionale orientate pe obiecte si, a datelor distribuite. Acest SGBDD, oferă un sistem complet de software pentru dezvoltarea rapida a aplicatiilor Internet.

Arhitectura multitier, reprezintă arhitectura cu mai multe niveluri (multitier) :

- unul sau mai multi clienţi ;
- unul sau mai multe servere de aplicaţii care executa parţi ale operaţiilor;
- un server de baze de date care stocheaza datele folosite de operaţii;
- Arhitectura sistemului permite distribuirea datelor pe maşini sau platforme diferite şi accesul partajat la acestea, prin intermediul aplicatiilor.

Arhitectura three-tier a sistemului Oracle



1.2.1 Oracle 9i Database

Serverul de baze de date este cheia rezolvării problemelor de administrare a datelor. Înfluenţa paradigmei client/server în domeniul bazelor de date a avut ca efect trecerea de la tehnologiile legate de partajarea fişierelor la aplicaţiile de reţea .

Astefel, în primul caz, accesul la o baza de date înseamnă transferarea pe client a executabilului SGBD-ului , cât şi a copiilor indecşilor fişierelor de date, rezultatul fiind trafic, considerabil în reţea , mecanisme slabe şi fragile pentru tranzacţii şi suport pentru concurenţa.

Serverele de baze de date au însemnat înlocuirea transferului de fişiere cu transfer de mesaje conţinând fraze SQL, în cazul cererilor clientilor şi date în cazul răspunsurilor, rezultatul fiind, reducerea traficului pe reţea, gestiune centralizata şi cresterea siguranţei datelor, mecanisme pentru asigurarea seriabilităţi tranzacţiilor.

Oracle 9i Database aduce noi funcţionaliţăti în aplicaţii cu baze de date, cât şi pentru aplicaţiile Internet Oracle 9i Database cum ar fi :

- Stocarea, regăsirea şi actualizarea datelor;
- Gestioneaza orice tip de informaţie;
- Descrierea datelor accesibile utilizatorilor;
- Oferă scalabilitate completă atăt pentru sistemele cu un singur procesor, cât şi pentru sistemele cu mai multe procesoare sau configuraţiile cluster cu mai multe noduri (baze de date distribuite);
- Suport pentru tranzacţii;
- Servicii de control pentru concurenţa;
- Servicii de autorizare a accesului de date (securitateea datelor), de comunicare, de integritate, pentru importul/exportul de date;
- Oferă posibilităţi de analiza a datelor;
- Permite implementarea rapidă a soluţiilor pentru afaceri etc…

Optiunea Oracle 9iReal Application Clusters(RAC), constituie o soluţie eficientă pentru imbunătaţirea bazelor de date distribuite.

Un Cluster este format din doua sau mai multe noduri (server-e) care executa tranzacţii în sistem concurenţial asupra bazei de date.
Tehnologia RAC permite cresterea numarului de tranzacţii posibile ,de adaugare de noi servere la un cluster.
Pentru crearea copiilor de siguranta, recuperare şi restaurare a fişierelor bazei de date, SGBD Oracle foloseşte instrumentul *Recovery Manager (RMAN)* – care este instalat în serverul de baze de date.
Bazele de date pot fi gestionate de catre sistemul de operare Windows 2000(20003) server pe care rezidă serverul de baze de date.Aceasta se face prin utilizarea Sistemului Oracle Managed Files(OMF).
Utilizarea acestui tip de fişiere :

- reduce consumul de spaţiu pe disc, prin ştergerea periodică a fişierelor nefolosite;
- înlatură problemele cauzate de specificarea greşită a fişierelor;
- simplifica procesul de creare şi testare a bazei de date (BD);

Oracle InterMedia asigura stocarea, gestiunea şi extragerea datelor de tip multemedia, servicii pentru localizarea acestora, servicii pentru Web etc.., cuprinde un set complet de componente *Audio Video,Image si Locator*.
Componentele Audio şi Video administreaza datele în format audio, video.Componeta Image permite stocarea, formatarea, regăsirea si conversia imaginilor prin tipuri de date obiect, prin intermediul obiectelor de dimensiuni mari BLOB şi, permite referirea imaginilor care rezida in fisiere externe (BFILE).
Componenta Locator permite codificarea geografica online pentru interogari şi aplicaţii ce folosesc obiecte de dimensiuni mari stocate în fişiere externe.
Java Server Pages(JSP) şi clasele Java Oracle InerMedia facilitează dezvoltarea aplicaţiilor Java, care pot incarca şi regăsi date în baze de date Oracle9i. Datele sunt stocate în obiecte recunoscute de Oracle InterMedia(ORAAudio, ORDDoc, ORDImage, ORDVideo.
Pentru eficientizarea administrării bazelor de date şi securităţii datelor în modelul relational, SGBD permite organizarea acestora in formate de cartuse de date (data cartridge), care este caracteristic unui anumit domeniu, care cuprinde date scalare şi structuri complexe de date (ex.multimedia).
Bazele de date Oracle 9i permit integrarea de cod Java în aplicaţii.Maşina virtula Java(Java Virtual Machine-JVM) este un program care interpreteaza surse Java, optimizat pentru diferite tipuri de platforme pe care se executa codul Java.Instalarea sistemului Oracle 9i cu opţiunea JVM, permite executarea aplicatiilor care utilizeaza proceduri stocate , SQLJ, CORBA/EJB, (Enterprise Java Beans), Servlet, JSP şi interfaţa Java Database(JDBC).

1.2.2 Oracle 9i Application Server(Oracle 9i AS) –oferă o infrastructura completă pentru dezvoltarea aplicaţiilor de tip e-business şi Internet.
Server-ul de aplicatii Oracle 9i include suport pentru Java (J2EE 1.3,1,4) şi serviciile Web (XML, XMI, SOAP, UDDI, WSDL, WebDAV).Oracle 9iAS asigură confidenţialitatea informaţiilor transmise prin retea, simplifica accesul la internet prin crearea de portaluri care oferă utilizatorilor un singur punct de accesare, fie prin browsere web, fie de pe dispozitive *Wirleess*, incluzând suporturi de criptare, autentificare şi autorizare,ofera un mediu de lucru eficient pentru dezvoltarea aplicaţiilor Web.
Platforma J2EE (Java 2 Platform,Enterprise Edition) defineşte un standard pentru dezvoltarea, implementarea şi desfaşurarea aplicaţiilor portabile şi scalabile, la nivel de companie, care este disponibil în trei variante : Java Edition, Standard Edition şi Enterprise Edition.Aceasta combina toate produsele cu suport J2EE într-un singur pachet, care include Oracle 9ias Containers for J2EE(OC4J), Oracle HTTP Server, Oracle9Ias TopLink, Jdeveloper si Enterprise Manager.

1.2.3 Oracle 9i Developer Suite

Oracle 9i Developer Suite extinde infrastructura formată din Oracle 9i AS şi Oracle 9i Database, care permite dezvoltarea de aplicaţii Internet sigure, scalabile şi fiabile.
Oracl 9i DS- reprezintă un mediu integrat de dezvoltare , care asigura intregul proces de creare a unei aplicaţii ,modelare, dezvoltare,analiza, codificare, depanare, optimizare, testare şi instalare.
Componetele Oracle 9i DS pot fi folosite pentru:

- gnerarea de servicii Web şi de aplicaţii;
- extinderea aplicaţiilor tranzacţionale tip business intelligence, interogari ad-hoc, rapoarte web;
- dezvoltarea rapidă de aplicaţii RAD (Rapid Application Development);
- creare de plicaţii de tip e-business,bazate pe internet.

*Clasa de utilitare Oracle 9iDS i*nclude componente pentru crearea de aplicatii Jdeveloper,Designer, Foms, Reports, Discoverer, Warefhouse Builder
Oracle 9i Jdeveloper- asigură un cadru de dezvoltare integrat cu Java, XML şi SQL, care permite crearea, depanarea şi instalarea rapidă a aplicatiilor de tip e-business şi a serviciilor Web ce pot rula pe orice sistem de operare.
Oracle XML Developer'Kit (XDK) este integrat în Jdeveloper, astfel încât utilitarul permite dezvoltatorilor Java să creeze, să prelucreze documente in format XML, de asemenea introduce doua modele UML, modelarea activitatilor si a claselor, care ofera um mod simplu de definere a proceselor de afaceri , respectiv dezvoltarea aplicaţiilor necesare pentru implemtarea acestor procese.
Oracle 9i Designer este un pachet de utilitare cu generatori integrati pentru proiectarea şi dezvoltarea de aplicaţii complexe, care permit:

- descrierea şi analiza modelului;
- proiectarea acestuia;
- generarea automata a bazei de ate;
- crearea de comenzi SQL pe baza diagramei entitate/relaţie;
- aplicaţii orientate pe obiecte.

Oracle 9iForms Developer este un utilitar rapid de aplicaţii (RAD) ce interacţioneaza cu bazele de date, este orientat pe evenimente, permiţând definirea prin PL/QL.
Oracle 9iSoftware Confiration Manager (SCM), permite crearea de aplicatii folosind echipe de dezvoltatori distribuite în diferite zone ale globului, în acest sens se asigura gestiunea eficientă a fisierelor, configurarea elementelor , arhitecturii aplicaţiei, stocarea şi administrarea tuturor fisierelor, directoarelor, datelor şi obiectelor.
Oracle 9i Reports este folosit pentru a crea rapoarte pe baza datelor din dictionarul de date.
Report Builder permite configurarea , publicarea şi distribuirea rapoartelor pe categorii de utilizatori.
Oracle 9i Business Intelliegence Beans, cuprinde o mulţime de standarde pe baza componentelor Jbeans, pentru implementarea rapida a aplicatiilor Java, componente OLAP.
*Oracle Discoverer e*ste un utilitar bazat pe interogari ad-hoc intuitive, rapoarte şi analize, oferă modalităţi de vizualizare a aplicaţiilor.
Oracle 9i Warehoue Bulder , este bazat pe modelul *data warehouse* şi pe conceptul business intelliengence, este folosit pentru organizarea de informaţii într-o baza de date.
Pentru dezvoltarea de aplicatii complexe, *SGBD ofera precompilatoarele Pro*(C, C++, PL/I ,Cobol,ADA, Fortran şi Pascal)* care permit incorporarea de instructiuni SQL sau blocuri PL/SQL in module scrie, folosind alte limbaj de programare.
Precompilatorul citeşte codul sursa şi generează un fişier ce poate fi procesat de către compilatorul respectiv.
Folosirea precompilatoarelor Oracle permite :

- dezvoltarea aplicaţiilor de baze de date;
- utilizarea tehnicilor de programare avansata;
- verificarea sintactică şi semantică a comenzilor şi a blocurilor PL/SQL;

- accesul concurent la baza de date distribuită Oracle din locaţii diferite ;
- specificarea opţiunilor de precompilare.

1.2.4 Oracle Enterprise Manager (OEM) este cel mai important utilitar de către sistem pentru administrarea bazei de date.

Arhitectura OEM constă din trei niveluri :

1. console client şi instrumente integrate ce furnizează o interfaţă grafică;
2. servere pentru gestiune şi un depozit de date;
3. agenţi inteligenţi .

Primul nivel al acestei arhitecturi reprezintă punctul central pentru lansarea utilitarelor Oracle, care permite admnistrarea completă a mediului, baze de date, aplicaţii, servicii. Diagnosticarea şi modificarea bazei de date, monitorizarea acesteia în reţea, administrarea nodurilor reţelei şi a serviciilor, pornirea instanţelor Oracle, personalizarea modului de afisare.

Al doilea nivel cuprinde un instrument Oracle Management Server(OMS), care gestionează schimbul de informatii între client şi agenţi inteligenţi, acesta foloseste un depozit Oracle Enterprise Management Repository pentru a stoca datele aplicaţiilor şi informaţiilor, respectiv a nodurilor.

Depozitul de date este format dintr-o mulţime de tabele care trebuie localizate intr-o bază de date Oracle, accesibilă prin OMS.

Al treilea nivel este compus din noduri care contin baze de date şi aplicaţii. La nivelul fiecarui Nod de monitorizare este instalat căte un agent intelligent.

Agentii inteligenti monitorizeaza evenimentele inregistrate, problemele apărute şi activităţile transmise de către un client prin nivelul de mijloc. Aceştia funcţionează independent de bazele date pe care le monitorizează, respectiv pot executa operaţii de oprire şi pornire a bazei de date.

Prin interfaţă grafică, *Oracle Enterprise Manager* permite :

- gestiunea instanţelor (instance menagement);
- efectuarea operaţiilor LDD;
- administrarea securităţii;
- administrarea spaţiului de stocare a bazei de date;
- executarea comnzilor SQL si PL/SQL;
- administrarea spatiului de lucru ,prin crearea de medii virtuale ;
- efectuarea exportului si a importului de date .

Alte componente ale Sistemului Oracle Pack, Tuning, oferă modalităţi de diagnosticare, optimizare şi administrare.

Oracle 9i Internet File System (Ifs) este o extensie a utilitarului OEM, care faciliteaza organizarea şi accesarea documentelor şi a datelor, servicii şi un depozit de date.

Oracle IFS acceptă protocoalele internet SMB, http, FTP, SMTP, IMAP.

1.2.5 Utilitare

SGBD Oracle oferă următoarele utilitare pentru administrarea bazei de date, asigurarea interfeţei de retea, accesarea datelor.

1) Oracle Universal Installer

Pentru instalarea, actualizarea sau dezinstalarea componentelor software şi crearea bazei de date implicite, SGBD oferă utilitarul Oracle Universal Installer(OUI), care este bazat pe un motor Java, respectiv are urmatoarele functii :

- detectează dependentele dintre componentele software;
- instalează componente software de la distantă, prin protocolul HTTP, folosind adresa URL a unei platforme de lucru;
- ţine evidenţa componentelor Oracle instalate pe o maşină;
- detectează limba naţională.

2)Oracle Database Configuration Assitant(ODCA)

Utilitarul ODCA reprezintă interfaţa grafică pentru utilizatori, care permit crearea, modificarea sau ştergerea bazei de date.
Cu ODCA se poate configura opţiunile bazei de date, de altfel permite adăugarea ulterioară , crearea ,modificarea unor template-uri ale BD,poate fi folosit pentru a crea BD cu o singură instanţă şi a creea instanţe intr-o configurare de tip *Real Application Clusters(RAC*).
*3)SQL*PLUS*-este cea mai folosită interfată pentru executarea comenzilor SQL şi a blocurilor PL/SQL.
Oracle 9i aduce o interfată de tip browser pentru SQL*PLUS, numita şi SQL*Plus, permite utilizatorilor să folosească un browser Web pentru a se conecta la bazele de date.
4)Utilitarele pentru import/export-permit transferul datelor din sau către o bază de date.
Utilitarul pentru export extrage definiţiile obiectelor şi tabelele cu date din baza Oracle şi le stocheaza intr-un fisier binar (Oracle Export File), acesta poate fi copiat prin ftp sau pe suport magnetic, iar transferul datelor din fişier în cealaltă bază se face prin utilitarul import.
*Sistemul Oracle 9i a a*dus următoarele funcţii pentru import/export:

- extragerea tabelelor dintr-un spatiu tabel;
- optimizarea statisticilor precalculate;
- introducerea unor parametri specifici de export/import.

*5)SQL*Loader,* oferă încărcarea datelor din fişiere externe (EX,C,C++) în tabelele bazei de date Oracle, care permite :

- Incărcarea datelor din mai multe fişiere
- Combinarea mai multor înregistrări de intrare intr-una singură, inregistrare logică de incarcare;
- Folosirea cîmpurilor de intrare cu lungime variabilă sau fixă;
- Prelucrarea datelor înainte de încărcare, folosind funcţii SQL;
- Generarea automată a valorilor coloanelor;
- Filtrarea datelor încărcate;
- Generarea de rapoarte;

*Utilitarul SQL*Loader* poate folosi următoarele fişiere;

- Un fişier de control care specifică formatul datelor încărcate;
- Fişiere de intrare care conţin date în formatul precizat;
- Un fişier de parametri (log file) care este creat de sql*loader
- Un fişier în care sunt scrise înregistrările respinse; (bad file), respectiv în care sunt stocate toate înregistrările ce nu satisfac criteriile de selecţie (discard file)
    - încarcarea coloanelor XML, a tabelelor şi coloanelor obiectelor (NCHAR,NVARCHAR,NCLOB).

9)Oracle Net Services-este o interfaţă de reţea , asigurată prin setul de utilitare *Oracle Net services, care conţine Oracle Connection* Manager, Oracle Listener, Oracle Configuration Assitant, Oracle Net şi Manager.
Oracle Net Configuration Assitant-este utilizat pentru configurarea componentelor de bazle retelei; listener, protocoale, metode de conectare, numele serviciului de reţea şi modul de utilizare pentru directory server.
O*racle Connection Manager* oferă transmiterea cererii de conexiune a clientului la serverul bazei de date.
Oracle Net este responsabil pentru stabilirea şi menţinerea conexiunii între o aplicaţie client şi serverul BD.
Oracle Listener rezidă pe server şi are responsabilitatea de a recepta cererea de conexiune a clientului.
10)DBVERIFY –este un instrument extern folosit pentru diagnosticarea bazei de date.
Acesta poate fi folosit pentru verificarea integrităţii structurilor fizice de date şi a fişierelor de recuperare a BD, verificarea segmentelor corespunzătoare unui tabel sau index.

1.3 Structura bazei de date ORACLE

Baza de date Oracle este o colecţie de date tratate unitar. Orice *baza de date* are o structură logică şi fizică.

1.3.1 Structura logică a bazei de date

Componentele structurii logice a unei baze de date Oracle sunt :

- blocul de date (block data);
- extensiile (extent);
- segmentele (segment);
- spaţiile tabel (tablespace);
- obiectele schemei (schema object).

Blocuri de date

Blocul de date reprezintă cea mai mică unitate logică I/O folosită în baza de date, corepunzatoare unui bloc fizic de octeţi de pe disc.

Dimensiunea blocului de date este definită în momentul creerii BD şi poate fi modificată ulterior.

Aceasta trebuie să reprezinte un multiplu al dimensiunii blocurilor fizice de la nivelul sistemului de operare.

Modificarea dimensiunii blocurilor logice de date se face prin intermediul parametrului DB_BLOCK_SIZE

Din punct de vedre structural, blocul de date Oracle cuprinde :

- un antet (header);
- un spaţiu liber (free space);
- un spaţiu pentru date (data space).

Antetul conţine informaţii generale despre bloc (adresa, tipul segmentului), un catalog al tabelelor (table directory) şi un catalog al liniilor (row directory)

Catalogul tabelelor conţine informaţii despre tabele care au date înregistrate, stocate în blocul respectiv, iar catalogul liniilor conţine informaţii despre liniile situate în bloc.

Spaţiul liber al blocului de date este alocat pentru inserarea de noi linii sau actualizarea liniilor care necesită spaţiu suplimentar, care depinde de valorile parametrilor **PCTFREE şi PCTUSED**

Parametrul PCTFREE reprezintă procentul minim din blocul de date care trebuie păstrat liber pentru actualizările deja exitente în bloc, valoarea implicită = 10 %.

Parametrul PCTUSED reprezintă procentul minim al spaţiului utilizat din bloc, care trebuie atins pentru a permite din nou inserarea unor linii de date, valoarea implicită = 40 %.

Spaţiul pentru date este format din linii de informaţii.

Fig.1.3.0 blocul de date

Informaţii antet
Catalog de tabele
Catalog de linii
Linii de date
Spaşiu liber

2)Extensii

Extensia este o unitate logica de alocare a spaţiului bazei de date, compusă dintr-o multime contiguă de blocuri de date, una sau mai multe extensii formează un segment.

O extensie este alocată atunci căND este creat sau extins un segment şi este dezalocată atunci căND segmentul este suprimat sau trunchiat.

Excepţii care au ca rezultat eliberarea extensiilor

- Proprietarul tabelului, sau un utilizator care are privilegiul **DELETE ANY** trunchiază tabelul sau gruparea folosind comanda T**RUNCATE….DROP STORAGE.**
- Periodic sistemul eliberează una sau mai multe extensii ale segmentului de revenire.

- Administratorul bazei de date DBA, eliberează extensiile neutilizate folosind comanda ALTER TABLE cu opţiunea DEALLOCATE UNUSED.

Parametrii care permit definirea dimensiunilor şi a liniilor extensilor în cadrul segmentelor sunt definiţi cu ajutorul clauzei STORAGE. Aceasta poate fi specificată în momentul creerii sau modificării obiectelor bazei de date (tabele, vizualizare, grupare, indecşi, partitii).
Clauza **STORAGE** are următoarea sintaxă :

STORAGE ({INITIAL intreg [{K|M}]
|NEXT intreg r[{K|M}]
|MINEXTTENTS intreg
|MAXEXTENTS {intreg r|UNILIMITED}
|PCTINCREASE intreg
|FREELISTS intreg
|FREELIST GROUPS intreg
|OPTIMAL [{intreg[{K|M}]|NULL]
|BUFFER_POOL {KEEP|RECYCLE|DEFAULT});

INITIAL - dimensiunea în octeţi a extensiei iniţiale,K=Kilobytes,M-megabytes
NEXT - dimensiunea în octeţi a următoarei extensii care va fi alocată segmentului.
MINEXTENS - reprezintă nr.minim de extensii, iar MAXEXTENTS – nr.total de extensii.
PCTINCREASE - procentul de creştere a dimensiunii unei extensii.
FREELISTS - nr.de componente ale fiecărui grup de liste libere pentru un tabel, o partitie, o grupare sau un index.
OPTIMAL - dimensiunea optimă în octeţi pentru segmentele de revenire.
BUFFER_POOL - permite specificarea unei zone de memorie cache

3)Segmente
Segmentul este un set de extensii care conţine toate datele unei structuri logice dintr-un spaţiu tabel. Segmentul foloseşte blocuri de date care se găsesc în acelasi spaţiu tabel.Baza de date conţine diferite tipuri de segmente

- segmente de date;
- segmente de index;
- segmente temporare;
- segmente de revenire.

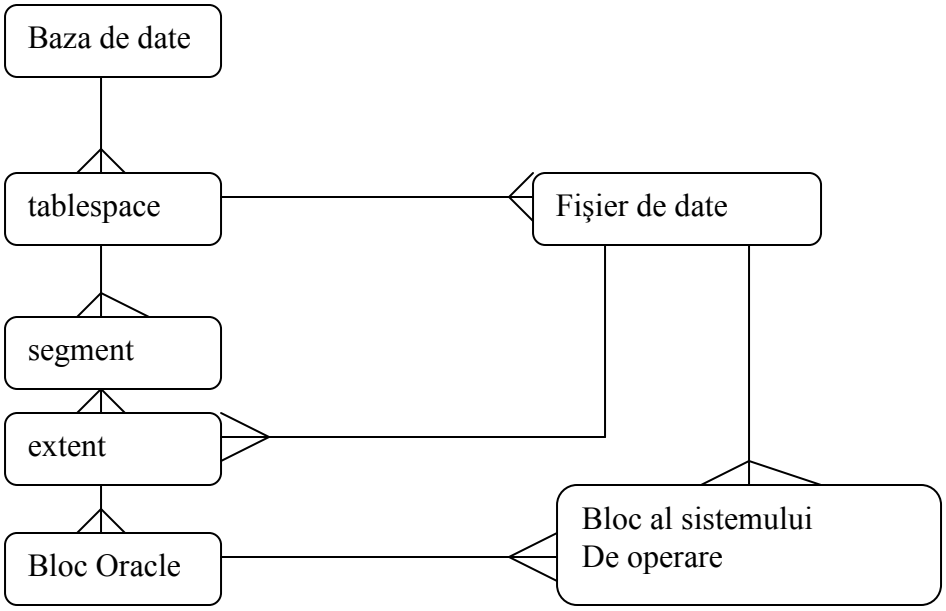


Fig.1.3.1 Diagrama E-R pentru relaţiile dintre structurile logice şi fizice de stocare

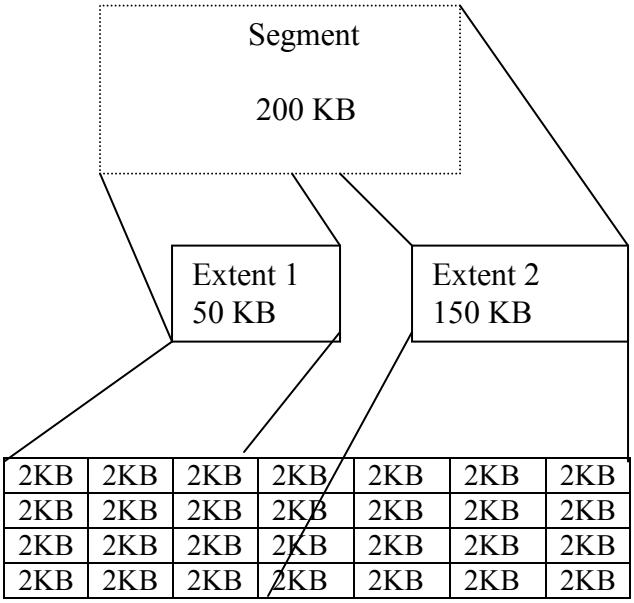


Fig.1.3.2 Formarea segmentelor din extent-uri şi a extent-urilor din blocuri de date.
Comenzile CREATE INDEX şi SELECT cu opţiunile ORDER BY, DISTINCT GROUP BY şi operaţiile UNION, INTERSECT, MINUS pot determina alocarea unui segment temporar

4) Spatii tabel

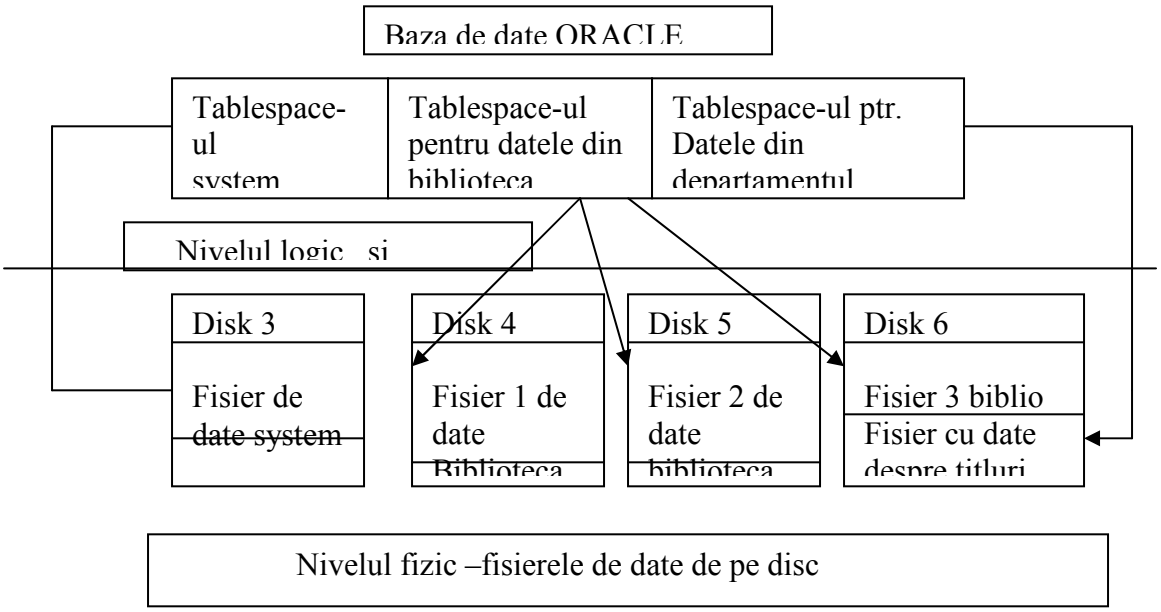
O bază de date este compusă din mai multe unităţi logice de stocare numite *spaţiu table.* Spaţiile tabel sunt utilizate pentru a grupa logic o mulţime de obiecte.
În orice bază de date Oracle, primul spaţiu tabel creat este SYSTEM, căruia va fi alocat automat.
Spaţiul tabel SYSTEM contine

- conţine dicţionarul datelor;
- segmentul de revenire;
- nu trebuie să conţina date ale utilizatorilor.

Spaţiile non-SYSTEM :

- Permit administrarea BD;
- Separă segmentele de revenire, segmentele temporare, segmentele de date şi segmentele index;
- separă datele dinamice de cele statice;
- controleaza spaţiul alocat pentru obiectele utilizatorilor.

Figura 1.3.3 tablespace –uri intr-o bază de date Oracle



Spaţiile tabel pot fi online (accesibile) sau offline.
Comanda prin care se crează un spaţiu tabel este **CREATE TABLESPACE.**
Forma generală
CREATE [UNDO]TABLESPACE nume_spatiu_tabel
|DATAFILE specificatie_fisier[clauza_autoextend][..
specificatie_fisier[clauza_autoextend]….]]
[minimum extent INTREG [{K|M}]]
[BLOCKSIZE intreg [k]]
[{LOGGING |nologging}]
[DEFAULT clauza_storage]
[{ONLINE|OFLINE}]
[{PERMANENT|TEMPORAY}]
[clauza_administrare_extensie]
[clauza_administrare_segment];

1.3.2 Structura fizică a bazei de date

Structura fizică a bazei de date Oracle presupune existenţa unui set de fişiere binare prin intermediul cărora se realizează stocarea fizică a datelor, aceste fişiere sunt împărţite în următoarele categorii :
a)fişiere de date (datafiles) ;
b)fişiere de rulare(redo log files) ;
c)fişiere de control(control files).

a)fişierele de date sunt fisiere fizice ale sistemului de operare. Acestea stochează toate datele tuturor structurilor logice ale BD.

Primul fisier de date creat este cel care stochează dicţionarul datelor, dimensiunea acestuia este cel putin 150MB

Caracteristicile generale

- dimensiunea fişierului poate fi schimbată ulterior creerii acestora şi poate fi extins pînă la o valoare specificată, atunci când spaţiul alocat este depăşit (alocarea dinamică).

O unitate logică numită spaţiu tabel (tablespace)

- un fişier de date poate fi asociat unui singur tabel şi unei singure baze de date.

Informatiile din fişierele de date pot fi accesate în timpul operaţiilor uzuale asupra BD în memoria cache.

Un fişier de date cu dimensiunea auto-extensibilă poate fi creat folosind una dintre comenzile *CREATE DATABASE,CREATE TABLESPAC, sau ALTER TABLESPACE* iar comanda *ALTER DATABASE* este folosită pentru a modifica modul de dimensoinare a unui fişier de date deja existent.

Alocarea unui fişier de date se face conform sintaxei următoare :

DATAFILE['nume-fisier'][SIZE intreg][{k|m}]][resuse][cluauza _autoextend]

Aceasta permite precizarea numelui şi dimensiunea fişierului.

În cazul spaţiilor tabel undo, pentru fiecare fişier trebuie specificată o dimensiune.

Dacă se foloseşte opţiunea auto_extended atunci forma generală este :

AUTOEXTEND{OFF|ON[NEXT
intreg[{K|M}]][MAXSIZE{UNILIMITED|intreg[[{K|M\}]}

Administratorul bazei de date DBA poate schimba dimensiunea unui fişier de date, folosind comanda **ALTER DATABASE.**

ALTER DATABASE[nume_baza]
DATAFILE['nume-fisier'][SIZE intreg][{k|m}];

Prin comandă ALTER TABLESPACE se pot adauga noi fişiere de date la un spaţiu tabel:

ALTER TABLESPACE nume_spatiu_tabel
ADD DATAFILE specoficatie_fisier[clauza_autoextended]………

În funcţie de tipul spaţiului tabel, sistemul oferă doua posibilităti pentru transferarea unui fişier de date, deci în ambele cazuri noul fisier trebuie să existe în sistemul de operare gazdă

Daca fişierul nu aparţine spaţiului tabel SYSTEM şi nu conţine segmente temporare sau de revenire ,atunci avem comanda :

ALTER TABLESPACE nume_spatiu_tabel
RENAME DATAFILE'nume_fisier'[,'nume-fisier'….] TO 'nume-fisier'[,'nume-fisier']

Daca fişierul apartine spatiului tabel SYSTEM sau altor spaţii tabel care nu poate fi niciodată offline, atunci este utilizată comanda *ALTER DATABASE,iar in aceasta situaţie BD, trebuie deschisă cu opţiunea MOUNT*

ALTER DATABASE[nume_baza]
RENAME FILE 'nume-fisier'[,'nume-fisier'……] TO 'nume-fisier'[,'nume-fisier'] ;

Prin interogarea vizualizărilor DBA_DATA_FILES şi V$DATAFILE din dicţionarul de date se pot obţine informaţii despre fişierele de date

b)Fişiere de rulare înregistreaza toate modificările care au loc asupra datelor bazei

O baza de date Oracle conţine doua sau mai multe fişiere de rulare,care asigura protectia BD în cazul defecţiunilor, vizualizările V$LOG şi V$LOGFILE din dicţionarul de date furnizează informaţii despre grupurile de rulare şi membrii acestora

c)Fişierele de control sunt fişiere binare de dimensiune redusă,necesare pentru pornirea şi funcţionarea BD

Interogând anumite vizualizări din dictionarul datelor se pot obţine urmatoarele informaţii despre fişierele de control :

- numele şi starea fişierelor asociate instanţei V$CONTROLFILE ;
- starea şi localizarea tuturor parametrilor V$PARAMETER ;
- inregistrările conţinute V$CONTROLFILE_RECORD_SECTION

- pentru a afla locaţia fişierelor de control poate fi utilizata comanda SHOW PARAMETER CONTROL_FILES ;

1.3.3 Dicţionarul datelor

În cele ce urmeza voi prezenta cel mai important instrument al bazei de date Oracle ,anume dicţionarul datelor

Aceasta include tabele şi vizualizări read-only ce conţin informaţii despre baza de date

- Definiţiile tuturor obiectelor din schema BD ;
- Cantitatea de spaţiu;
- Valorile implicite ale coloanelor ;
- Constrăngerile de integritate ;
- Numele utilizatorilor ;
- Privilegiile şi rol-urile acordate fiecarui utilizator ;
- înformaţii de auditare ;

Dictionarul de date este generat automat la crearea bazei de date şi este reactualizat de catre server,iar continutul său reflectă imaginea bazei de date (structura fizica si logica) la un moment dat

Dicţionarul de date contine tabele şi vizualizari publice ,respectiv numit *DUAL* ,pe care aplicaţiile îl pot referi pentru a garanta un rezultat cunoscut.

Vizualizările decodifica înformaţiile stocate in tabelele de baza .Aceste vizualizari sunt create

Utilizând script-ul *catalog.sql,*care trebuie rulat de utilizatorul SYS cu privilegiul SYSDBA

Vizualizarile dicţionarului pot fi relative la:

1. toate obiectele din BD (cu prefix DBA_)
2. obiectele accesibile unui utilizator (ALL_)
3. obiectele unui utilizator (USER_)
4. perfomanţe (V$ sau GV$)

În funcţie de valorile sufixului (precizat între paranteze) ,furnizează înformaţii despre :

- utilizatori (USER) ;
- coloane specificate în constrăngeri(CONS_COLUMNS) ;
- tabele ale BD(TABLES) şi tabele externe (EXERNAL_TABLES) ;
- vizualizări (VIEWS) şi vizualizări materializate(MVVIEW) ;
- grupari(CLUSTERS) şi indecsi(INDEXES) ;
- declanşatori(TRIGGERS) ;
- sinonime (SYNONIMS) şi secvenţe(SEQUENCES) ;
- obiecte(OBJECTS) ;
- biblioteci (LIBRARIES) ;
- politici de securitate(POLICIES) ;
- privilegii obiect asupra tabelelor(TAB_PRIVS) ;
- tipuri obiect(TYOPES) şi colecţie (COLL_TYPES);

Vizualizarea DBA_OBJECTS este necesara pentru obţinerea de informaţii referitoare la toate categoriile de obiecte ale unei scheme

Coloanele sunt urmatoarele :

1. OWNER –proprietarul obiectului ;
2. OBJECT_NAME-numele obiectului;
3. SUBOBJECT_NAME-numele subiectului;
4. OBJECT_ID- identificatorul obiectului;
5. DATA_OBJECT_ID –identificaorul segmentului ;
6. OBJECT_TYPE-tipul obiectului;
7. CREATED-data creării obiectului;
8. LAST_DDL_TIME-data ultimei modificări;

9. TIMESTAP-specificarea formatului datei;
10. STATUS-starea obiectului;
11. TEMPORARY-precizarea că obiectul este temporar;
12. GENERATED-specificarea modului de generare a obiectului;
13. SECONDARY –precizarea obiectului secundar;

Exemple :

1.Să se afiseze proprietarul ,numele şi tipul obiectelor din BD;

```
SQL>SELECT OWNER,OBJECT_NAME,OBJECT_TYPE
FROM SYS.DBA_OBJECTS;
```

2.Să se afişeze proprietarul,numele şi tipul tuturor obiectelor accesibile utilizatorului curent ;

```
SQL>SELECT WNER,OBJECT_NAME,OBJECT_TYPE
FROM  ALL_OBJECTS;
SELECT WNER,OBJECT_NAME,OBJECT_TYPE
FROM  USERR_OBJECTS;
```

3.Să se obţină identificatorul şi starea sesiunilor iniţiate de utilizatorul elev

```
SQL>SELECT SID,STATUS
FROM V$SESSION
WHERE USERNAME=’elev’;
```

listing create_biblio_tablespace.sql

```
-- conectarea la  SYSTEM –se introduce valoarea pentru  1 ,2 (biblio)
spool biblio_createTablespaces.log
--connect sys/&&1@&&2 as sysdba
set echo off
set termout on
set verify off
define tmp1=_data;
define tmp2=_indx;
define tmp3=01.dbf;
create tablespace &&1&tmp1
        datafile '&&2\&&1&tmp1&tmp3' size 50M reuse
        autoextend on next 1280K
        minimum extent 128K
        default storage ( initial 128K next 128K minextents 1 maxextents unlimited
pctincrease 0 );
create tablespace &&1&tmp2
        datafile '&&2\&&1&tmp2&tmp3' SIZE 20M REUSE
        autoextend on next 1280K
        minimum extent 128K
        default storage ( initial 128K next 128K minextents 1 maxextents unlimited
pctincrease 0 );
undef tmp3;
undef tmp2;
undef tmp1;
spool off
exit
```

Capitolul 2

Realizarea bazei de date in Oracle (sql*plus) si in visual foxpro 9.0

2.1 deschiderea si inchiderea unei sesiune de lucru in SGBDD Oracle 9i

2.1.1 deschidera sesiuni de lucru SQL*PLUS

Oprima conectare se face atunci cand se incepe lucrul cu SQL*PLUS

Din meniul principal al aplicatiei se selecteaza optiunea SQL*Plus, dupa care apare o fereastra in care se introduce nume utilizator, parola, nume – serviciu

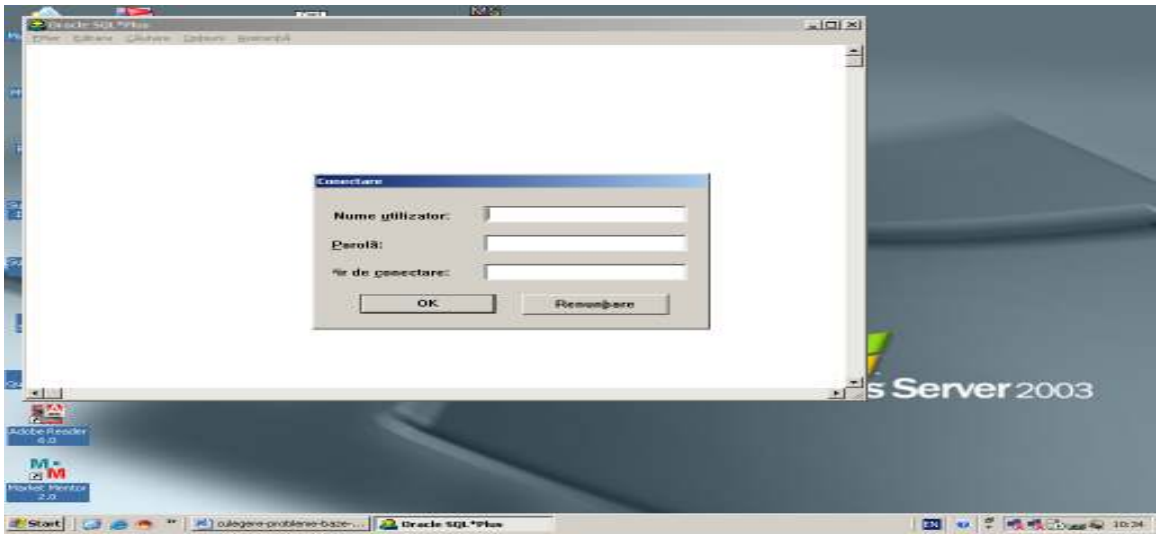
Sintaxa generala :

Sql>sql [nume-utilizator/parola]
[@nume_baza_de_date][nume_fisier][-silient][sqlplus/nolog]

Unde :

Nume-utilizator si parola sunt numele unui utilizator cu drept de acces la baza de date, respectiv parola introdusa la configurarea sa

@nume_baza_de_date este numele baze de date cu care se lucreaza in retea



Exemplu 1 Se introduce nume-utilizator scott, parola : tiger

Ex 2.2 sql>sqlplus biblioteca/bib @biblioteca

2.2.2 Conectarea utilizatorului la o baza de date Oracle

Sintaxa este :

CONN[ECT] [nume-utilizator][/parola] [@nume_baza_de_date] ;

Ex 2 Sa se conecteze la baza de date biblio (preluarea cartilor din biblioteca)
SQL>CONN BIBLIO/biblio@biblio;
Connected.
Unde :biblio este baza de date oracle
2.1.3 Deconectarea utilizatorului de la baza de date (BD)
Deconectarea se face prin comanda DISCONNECT
Sintaxa este :
Sql>DISC[ONNECT] ;
Limbajul de programare Oracle permite primele patru caractere de lucru
Ex 3 Sa se deconecteze de la BD biblio ;
Sql>DISC ;
Si apare mesajul de deconectare Disconnected from ORACLE.
Sau
Sql>DISCONNECT ;
2.1.4 Inchiderea SGBDD Oracle se face folosind comanda QUIT, sau EXIT sau ^Z
SQL>QUIT ;
sau
SQL>EXIT ;
sau
SQL>^Z

2.1.5 Crearea legaturii cu baza de date de la distanta
SGBD Oracle permit atat lucrul cu o baza de date locala cat si lucrul in retea
Pentru crearea unei legaturi intre baza de date locala si o baza de date aflata la distanta se utilizeaza comanda CREATE DATABASE LINK cu urmatoarea sintaxa :
CREATE [PUBLIC] DATABASE LINK nume-legatura
CONNECT TO nume-utilizator identified by parola
USING ‘baza-de date-la distanta’;
Unde :
Optiunea PUBLIC face ca legatura creată să fie publică, altfel va fi privată
Exemplu 5 Să se creeze o legătură privată la baza de date cu numele global vanzari.romanaia.ro
Utilizatorul biblioteca , parola biblio

Sql >CREATE DATABASE LINK vanzari.romania.ro
Connect to biblioteca identified by biblio;
Odata create legatura, utilizatorii pot accesa datele bazei de date respective prin folosirea numelor globale ale obiectelor
O legatura globala se creeaza cu ajutorul produsului Oracle Name Server
Exemplul 6 Sa se selecteze toate inregistrarile autori din serverul bazei de date Biblio

Sql>SELECT * FROM autori@vanzari.romania.ro;

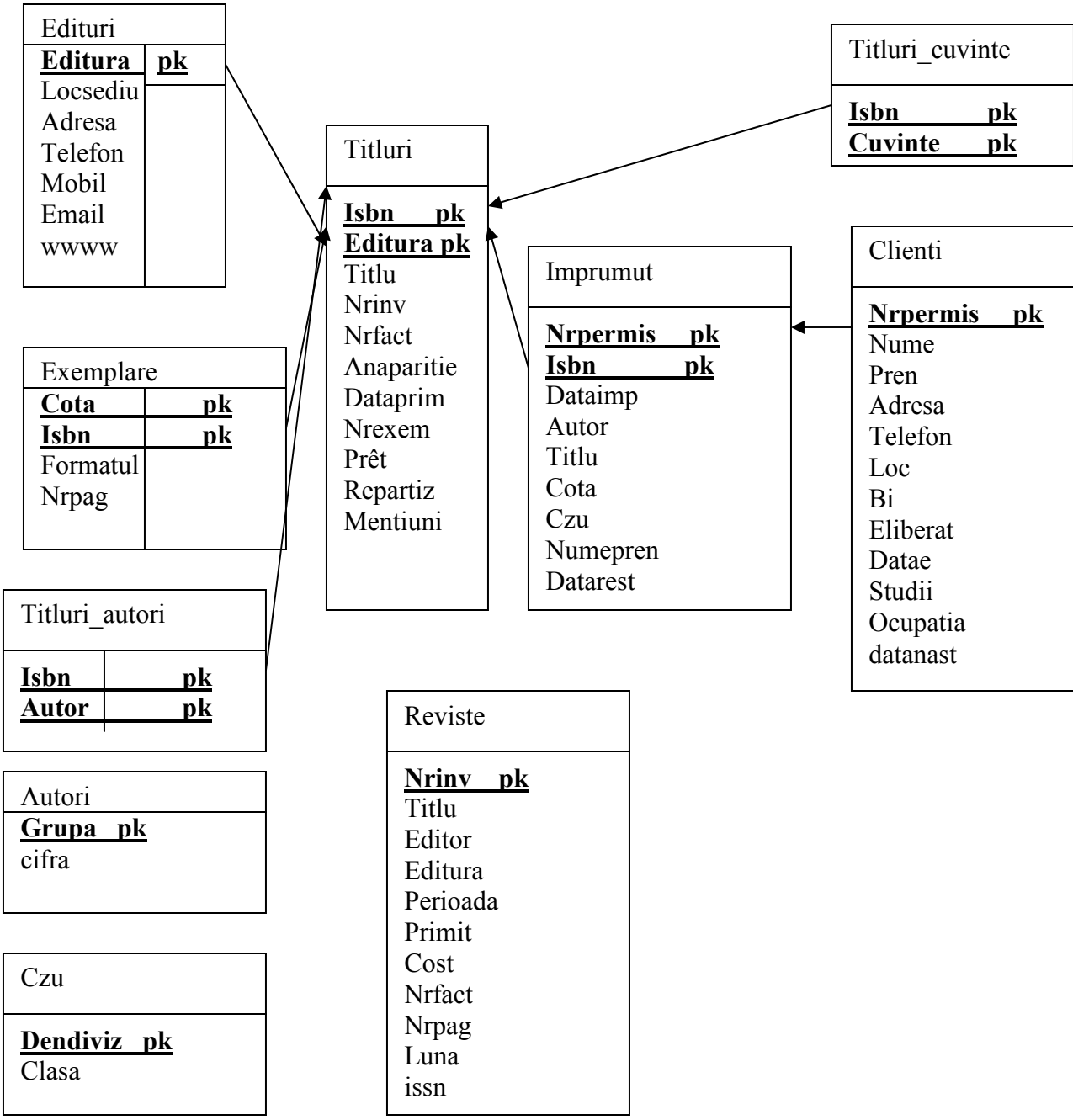
Exemplu 7 Sa se sterga legatura create anterior

Sql>drop [public] database link vanzari.romania.ro ;

 ISSN 2069-7988
ISSN-L 2069-7988

2.1.6 crearea si actualizarea structurii tabelelor si definirea restrictiilor 1.prezentarea schemei bazei de date Biblioteca

Pasul 1 Schema bazei de date BIBILOTECA
1)Prezentarea schemei bazei de date BIBLIOTECA



1)Tabela Autori –conţine informaţii despre nomenclatorul de autori –numită tabela de autori a lui Cutter Ch .Numarul din tabela, corespunzator primei sau primelor silabe (grupe de litere) a numelui de autor sau titlului puclicatiei, se adauga la initiala autorului sau primului cuvant din titlului, se obţine semnul de autor
Tabela de autori are urmatoarea structura :

- Grupa,c,9 –reprezinta grupa de autori, fiind si cheia primara
- Cifra,n,2 –cifra corespunzatoare tabelei

Exemplu

Grupa	Cifra	Grupa
A	10	B

Aa	11	Ba
Abe	12	Bac

Exemple de codificare semn autor

2)Tabela CZU (Clasificarea zecimala universala)-contine totalitatea cunostintelor umane ca o singura unitate imparţita in zece mari clase, notate cu cifra arabe, de la 0-9, iar fiecare clasa se imparte in 10 subdiviziuni etc…..

Cota se noteaza pe fisa cu o linie orizontala desparţitoare la mijloc şi se completeaza in colţul stanga-sus, iar in colţul stanga –jos indicele CZU, care indica locul fisei in catalogul systematic

Tabela CZU are urmatoarea structura :

-dendiviz,c,35-subdiviziunea carti-fiind cheia primara, fără spatii la inceput de text

-clasa,c,15-reprezinta clasa

Exemplu

Dendiviz	Clasa
Generalitati	0
Informatica	51
Comert.relatii economice internationale.	339

3)Tabela EDITURI-reprezinta nomenclatorul editurilor de publicaţii având urmatoarea structura de baza

- Editura,c,30-reprezinta editura (de exemplu:ALL,POLIROM,TEHNICA ETC….), care reprezinta si cheia primara
- Locsediu,c,30-locul sediu al editurii (ex:Bucuresti,Iasi….)
- Adresa,c,30-adresa completa a editurii (strada,bloc,nr,ap etc….)
- Telefon,c,10-reprezinta telefonul fix
- Mobil,c,10-telefonul mobil;
- Email,c,30-adresa de e-mail al editurii
- Wwww,c,30-adresa de internet al editurii

Editura	Locsediu	Adresa	Telefon	Mobil	Email	Wwww
All	Bucuresti	b-dul Timisoara	0214130715	0724123456	edit@all.ro	All.ro

4)Tabela Comenzi contine informatii despre comenzile facute de catre o biblioteca,avand structura:

- Editura,c,30-reprezinta editura de publicatie
- Autor,c,35 –autorul cartii
- Titlul,c,60 –titlul publicatiei
- Repartiz,c,20-reprezinta repartizarea cartilor conform structurii,anume:CZU1/3,902/904,908,93/99-Filosofie,czu 5/6,91-stiinte exacte, tehnica, geografie, czu 8-lingvistica, czu 0,7,929-generalitati etc….
- Mentiuni,c,30-mentiunea cartii
- Datacom,date-data comenzii
- Nrex,n,4-numarul de exemplare oferite/cerute

5)Tabela TITLURI –contine informatii despre cartile intrate/iesite /interschimbate intr-o biblioteca, având urmatoarea structura de baza:

- ISBN,C,13-NOT NULL-reprezinta numarul standard international pentru carti, fiind şi cheia primara de exemplu (ISBN=973-9392-46-6ETC…), iar pentru reviste, ziare periodice se utilizeaza ISSN

- TITLUL,C,60-titlul cartii (de exemplu:sistem de gestiune a bazelor de date – aplicatii)
- Nrinv,n,6-reprezinta numarul de inventar scris pe carte
- Nrfact,n,15-reprezinta numarul documentului de intrare (factura,aviz,etc….)
- Anaparitie,n,4-anul aparitiei ☹ex:2005,2006)
- Prêt,n,12,2-pretul cartii
- Dataprim,date-data primirii cartii in biblioteca sub forma zz.ll.aaaa
- Repartiz,c,20-repartizarea dupa continut ,vezi tabela de comenzi
- Mentiuni,c,20-la fel;
- Nrexem,n,6-reprezinta numarul de exemplare intrate/iesite/interschimbate intr-o biblioteca
- Editura,c,30-editura care reprezinta o cheie straina pentru tabela edituri

6)Tabela Exemplare-contine date despre cota topografica ,respectiv formatul care are urmatoarea structura:

- Cota,c,15,not null –cotoa topografica prezentata in tabela clasificarea zecimala univcersala(CZU), fiind si cheia primara
- Formatul c,15-formatul (inaltimea publicatiei)-asezara cartilor in depozite pe rafturi,avand urmatoarele informatii :
  - -Formatul I-<20cm
  - -Formatul II –de la 20 cm-la 25 cm
  - -Formatul III-de la 25-la 30 cm
  - -Formatul IV –de la 30-38 cm
  - -Formatul V peste 38 cm

Marcat in cota prin cifre romane de exemplu I 3214, V 324-formeaza cota topografica, pentru diferite categorii de tipărituri se utilizeaza alte litere P(periodice),M(muzica),H(harta),Ars(arta)etc…..

- Nrpag,n,4-inseamna numarul de pagini
- Isbn,c,13 numarul standard international penru carti-fiind cheia starina

7)Tabela titluri_autori –contine informatii despre autorii publicaţiei ,având structura:

- Isbn,c,13-not null –cheie straina decalarata prin references titluri tag isbn;
- Autor,c,35-reprezinta autorul cartii –not null cu mentiunea ca prima litera trebuie sa fie masjuscula si fara spatii la inceput
- Cheia primara fiind isbn+autor

8)Tabela titluri_cuvinte-contine informatii despre cuvintele cheie ale cartilor,anume: (bazae de date relationale,aplicatii distribuite realizate cu Java…),cu structura:

- Isbn,c,13
- Cuvinte,c,30-reprezinta cuvintele cheie ale publicatiei
- Cheia primara=isbn+cuvinte

9)Tabela Clienti-reprezinta tabela cu informatii despre inscrierea cititorilor intr-o biblioteca,cu urmatoarea structura:

- Nrpermis,n,6-numar permis de intrare care este unic,fiind si cheia primara a tabelei
- Nume,c,25;pren,c,15-numele /prenumele cititorului (elevului din cadrul scolii/profesorului)
- Adresa,c,30-adresa completa conform actului de identitate
- Telefon,c,10-numar de telefon fix,mobil,acasa,scoala,serviciu;
- Loc,c,15-locatitata
- Bi,c15-seria buletinului de identitate si seria

- Eliberat,c,15-date,date-data eliberarii
- Studii,c,15-studii {elev,student,……}
- Ocupatia,c,25-ocupatia de baza a profesorului,elevului etc…

10)Tabela imprumut de carti –reprezinta fişa centrala de contact de imprumut de carţi catre cititori inregistrati,avand structura de baza:

- Nrpermis,n,6-fiind cheia primara de la tabela clienti,daca nu exista in baza de date atunci nu se poate imprumuta o carte
- Nrinv,n,6-numar inventar inregistrat pe carte
- Dataimp,date-data imprumutului sub forma zz/ll/aaaa
- Autor,c,35-autorul cartii de imprumutat
- Titlu,c,60-titlul cartii de imprumutat
- Czu,c,20-indicele de cota prezentat in tabela CZU
- Cota,c,15-cota topografica prezentata anterior
- Numepren-numele /prenumele cititorului adaugat autmat din tabela clienti
- Datarest, date (zz/ll/aaaa)-reprezinta data restiturii cartii, fiind atributul cel mai important ,care o conditie de 15 zile –necesara in stabilirea listei restantelor de carti , conform fisei cartilor
- Cheia prima este formata din str(nrpermis,6)+str(nrinv,6)

11)tabela de vederi :fisa cartii,fisa de imprumut,lista restantelor,reviste,ziare
Fisa de catalog are dimensiuni de stas international,iar in lipsa lor se folosesc listele obisbuite listate la imprimanta

Fisa de catalog

Descrierea bibliografica prezinta elementele esentiale de identificare a publicatiei:titlul, autorul/autorii sau responsabili de lucrare,editia, date de publicare (loc,editura,an), descrierea cantitativa(paginatia), colectia, alte note si numarul standard (ISBN) pentru carti si ISSN pentru periodice
In afara de aceste elemente de descriere bibliografica, pe fiecare fisa se stabileste cate o vedeta,vedeta este cuvintul care da ordinea de intrare a fisei in catalog si care se scrie pe primul rand de la prima verticala
In vedeta,numele autorilor personali se scriu mai intai cu numele si apoi cu prenumele
Schema generala este:
VEDETA
Titlul:informatii la titlu/mentiune de responsabilitate(autorii asa cum sunt trecuti pe carte),-editia.
-loc publicare:editura,anul publicarii.
Paginatie:ilustratii.-(colectia;numarul in cadrul colectiei).
Note.
I.S.B.N
Punctuatia din schema de mai sus trebuie respectat intocmai:
Exemplu:
Programare avansata in Oracle 9i/Ileana Popescu,Alexandra Alecu,Letita Velescu,Gabariela Florea
Bucuresti:Editura Tehnica,2004
Bibliografie.
ISBN 973-31-2208-4
I..Popescu,Ileana
II..Alecu,Alexandra
III.Velescu,Letita

IV.Florea,Gabriela

004.42 ORACLE9i

2.Tipuri de date in Oracle

Clasificarea tipurilor
a)tipuri predefinte
-scalare
-Colectii
-refreinte
b)tipuri de date definite de utilizator
a)tipuri de date scalare predefinite
Principalele tipuri de date scalare de date in Oracle

Tip	Descriere	Comentarii
CHAR(n)	Sir de caractere cu lungime fixa egala cu n octeti	Lungimea fixa presupune completarea automata cu spatii a valorilor cu lungime mai mica Lungimea maxima=2000 bytes
VARCHAR2(n)	Sir de caractere cu lungime variabila	O valoare de lungime mai mica a atributului la un moment dat nu va fi completa cu spatii Lungimea maxima=4000 bytes
NUMBER(p,s)	Date numerice de lungime variabila	P=precizia (lungimea maxima a numarului) S=scala(nr.de pozitii zecimale rezervate P=maxim=38 cifre
DATE	Date calendaristice si temporale	Lungime fixa de 7 bytes
CLOB	Caracter Large Object	Poate stoca sisuri de caratere pana lsa 4GB Un atribut de acest tip nu poate fi folosit in suniterogari, functii sau clauze WHERE a unei fraze SELECT SQL
LONG	Date de tip sir de caractere de lungime variabila	Poate stoca sisururi de caractere pana la 2 gb la fel ca la CLOB
Blob	Date binare nestructurate	Maxim 4 gb
Bfile	Date binare stocate intr-un fisier extern	Pointer spre un fisier de pe disc cu dim=4gb
Rowid	Date in format binar	Acest tip este specific

	reprezentand adresa fizica pe disc a inregistrarii	psudocodului rowid

Obs.pentru un sir de caratere este apostroful nu ghilimele

Exemplu Sa se creeze tabela test
Sql>CREATE TABLE test (tsdata DATE);
Adaugarea unei inregistrari
Sql>INSERT INTO TEST VALUES('10-OCT-06');
sau folosinf functiile de conversie
SQL>INSERT INTO TEST VALUES(TO_DATE('10/10/2006','DD/MM/YYYY');

c)Tipuri abstracte definite d utilizator
Sintaxa generala este :
CREATE TYPE dentipAS
Object(ATRIB_1 TIP,ATRIB_2 TIP,.....)
Exemplu 2 Sa se tip adresa_ty pentru definirea persoanelor care locuiesc intr-o localitate
CREATE TYPE ADRESA_TY AS OBJECT(
STRADA VARCHAR2(301),
numar varchar2(10),
bloc varchar2(10),
scara varchar2(10),
ap integer);

crearea tabelei locuitori
create table locuitori(
cnp integer,
nume varchar2(35),
adresa adresa_ty);

exemplu 3 inserarea de articole in tabela locuitori
SQL>INSERT INTO Locuitori values(1000,'Aurel v', adresa_ty('b-dul bucegi','10bis','b1','sc1',10));
extragerea de informatii din tabele este de forma aliastabela.atributtabela.atributtip
EXEMPLU 4
SQL>SELECT nume L.ADRESA.STRADA FROM LOCUITORI L WHERE CNP=1000;
EXEMPLU 5 Sa se sterga tipul creat anterior
SQL >DROP TYPE adresa_ty
Obs.este necesara folosirea unui alias (sinonim) pentru resusita si actualizarea atributelor, altfel vom obtine un mesaj ORA-00904 INVALID COLUMN NAME
Un tip utilizat de o tabela nu poate fi sters!

 ISSN 2069-7988
ISSN-L 2069-7988

d)Colectii
Colectiile sunt seturi de date ce pot fi trate ca parte a unei singure inregistrari dintr-o tabela
Clasificare

- Tipuri de date vectori cu marime variabila (varing arras)
- Tabele incapsulate (nested tables)

Un vector cu marime variabila se declara astfel:
CREARE OR REPLACE TYPE<TIP> AS VARRAY(12) OF <TIPELEMENT>

Unde tipelement poat fi scalar sau abstract
Exemplu 5 Vom considera o tabela personal din cadrul bibliotecii, al carei atribut zilelucrate va stoca in fiecare inregistrare un vector de 12 elemente pentru numarul zilelor lucrate

```
CREATE OR REPLACE TYPE zile_ty as varraay(12) of integer
/
Create table personal(
Nume varchar2(35),
zilelucrate zile_ty);
```

inserarea se face astfel:

```
insert into personal values('Amarie',zile_ty(22,21,20,22,23,null,null));
```

extragerea informatiilor necesita utilizarea functiei table() ce preia ca argument atributul de tip vector
Exemplu 6

```
SQL>SELECT column_name,data_type,data_length,data_precision,data_scale
From user_tab_columns
Where table_name='personal'
/
```

Exemplu 7 Sa se listeze informatiile din tabela Personal folosind clauza From pentru campurile nume,zilelucrate
Exemplu 8 Sa se modifice tabela pentru nume='Amariei' trecand toate zile lucratoare dintr-un an de lucru

```
SQL>UPDATE PERSONAL
SET zilelucrate=zile_ty(20,21,18,22,20,22,20,22,20,22,20,21,22,22)
Where nume='Amariei'
```

Tabele incapsulate reprezinta un alt tip de colectie care poate stoca un numar nelimitat de elemente si ofera posibilitatea actualizarii directe a unui element
Sintaxa este urmatoarea :
CREATE OR REPLACE TYPE<tip>AS TABLE OF<tipelement>
Exemplu 9 Sa se creeze tabela incapsulata pe baza unui tip abstract si care utilizeaza un atribut al unei tabele clasice

```
CREATE OR REPLACE personal_ty AS OBJECT(
Nume varchar2(35),
salariu number(12,2)
```

```
)
/
Create type personal_nt as table of personal_ty
/
Create table biblioteca(
Dencomp varchar2(35),
personal personal_nt
)nested table personal store as personal_nt_tab;
```

obs. In definirea tabelei biblioteca vom observa o extensie a instructiunii create table :
NESTED TABLE<numeatribut>STORE AS<numetabeldestocare>
Aceasta extensie se datoreza faptului ca elementele atributului personal sunt stocate sub forma unei tabele clasice
Exemplu 10 Sa se adauge articole in tabela personal

```
SQL>/
INSERT INTO PERSONAL(nume) values('AAAAAAAAAAA')
*
ERROR at line 1:
ORA-01400:cannot insert NULL into ("AAAA"."PERSONAL".NUME")
```

Pentru rezolvare se utilizeaza secventele
Adaugarea unor inregistrari noi in tabela Biblioteca se utilizeaza notatia :
numetiptabelaincapsulata(numetipelement(valatribut1,.....))
exemplu 11 Sa se adauge doi salariati in compartimentul Bibliotecar

```
INSERT INTO BIBLIOTECA VALUES('Bibliotecar',
personal_nt(
personal_ty('Apostu',200000000),
personal_ty('Barbu V',50000000)
)
);
```

Extragera datelor se face folosind functia THE cu formatul
THE(Select<atributtiptabla>FROM<tabelaparinte>)
Ce are rolul de "aplatiza" inregistrarile din tabela incapsulata si de a le prezenta sub forma de linii ca rezultat al interogarii tabele parinte

```
SQL>desc user_constraints;
```

se afiseaza interogarea tabelelor ce contin atribute de tip tabela incapsulata
exemplu 13 Sa se interogheze dupa valorile atributelor tipului abstract personal_ty stocat in tabela incapsulata

```
SQL>DESC USER_COLUMNS;
```

exemplu 14 Sa se adauge inregistrari in tabela incapsulata folosind functia THE

```
sql>insert into the(select personal from biblioteca where dencomp='Bibliotecar') nt
values(persoanal_ty('Amariei ',2000000));
```

3.Formatul simplu al comenzii SQL CREATE TABLE .Valori implicite

Exemplu 15 Sa se creeze tabela personal_biblioteca

```
SQL>CREATE TABLE PERSONAL_BIBLIOTECA(
Marca integer,
```

nume varchar2(35),
compartiment varchar2(5),
datasv date,
salaorar number(12,2),
);
In dictionarul de date Oracle, o parte dintre informatii se gasesc in tabela sistem user_tables
Exemplu 16 Sa se afle numele tutoro tabelelor din baza de date
Pentru aceasta folosim interogarea SQl.>SELECT table_name
From user_tables;
o alta tabela virtuala a dictionarului din care pot fi extrase numele tabelelor este USER_OBJECTS;
EXEMPLU 17
SQL>SELECT object_name
From user_objects
Where object_type='TABLE';
Pentru a afla detalii despre atributele unei tabele este necesara consultarea unei alte tabele a dictionarului de date – USER_TAB_COLUMNS
SQL>INSERT INTO PERSONAL_BIBLIOTECA(MARCA,NUME)
VALUES(100,'AAAA');
1 ROW CREATED.
SQL>INSERT INTO personal_biblioteca(marca,nume) values(100,'AAAA');
APARE EROAREA
*
ERROR at line 1:
ORA-00001:UNIQUE
CONSTRAINT(APOSTU.PK_PERSONAL_BIBLIOTECA)VIOLATED.
Exemplu 18 Sa se afiseze informatiile despre tabela personal_biblioteca
In conformitate cu stantardele SQL, declarea valorilor implicite pe care le poate lua un atribut , l inserarea unei linii se utlizeaza clauza DEFAULT.

Exemplu 19 Sa se creeze tabela personal_biblioteca CU CLUAZA default
SQL>CREATE TABLE PERSONAL_BIBLIOTECA(
Marca integer,
nume varchar2(35),
compartiment varchar2(5) default 'BIBLIO',
datasv date DEFAULT SYSDATE,
salaorar number(12,2) DEFAULT 45000,
);

EXEMPLU 20 Sa se creeze o copie a tabelei personal_biblioteca in tabela copie_PERSONAL
CREATE TABLE copie_pers AS
SELECT * FROM PERSONAL_BIBLIOTECA;
Noua tabela va prelua toate atributele din personal_biblioteca, nu insa si restrictiile
In Oracle9i se poate redenumi o tabela
Exemplu 21 Sa se redenumeasca tabela copie_pers in pers_ian2006;

SQL>RENAME COPIE_PERS TO PERS_IAN2006;

4.Valori nunule.Cauza NOT NULL
EXEMPLU 22 Sa se creeze tabela personal_biblioteca cu cluaza NOT NULL

SQL>CREATE TABLE PERSONAL_BIBLIOTECA(
Marca integer NOT NULL,
nume varchar2(35) NOT NULL,
compartiment varchar2(5) default 'BIBLIO' NOT NULL,
datasv date DEFAULT SYSDATE,
salaorar number(12,2) DEFAULT 45000,
);

In cazul in care nu trecem clauza NOT NULL la inserarea unei linii apare o eroare de tipul
ERROR at line 1:
ORA-12991:column is refrenced in a multi-column constraint
Ceea ce reprezinta incalcarea restrictiei de nenulitate
Exemplu 23 Sa se afle posibilitatea sa imposibilitatea primirii de valori nule pentru fiecare atribut al tabelei personal_biblioteca
SELECT column_name,nullable
From user_tab_columns
Where table_name='PERSONAL_BIBLIOTECA';

5 Chei primare si alternative.Clauzele PRIMARY KEY si UNIQUE
Cheia primara a unei relatii este cea care identifica fara ambiguitate oricare linie a tabelei, este declarata prin clauza PRIMARY KEY
Pentru atributele de tip cheie candidat se poate folosi clauza UNIQUE care va respecta unicitatii valorilor
Exemplu 25 .Sa se creeze tabelele de mai jos utilizand clauza unique si not null,primar key
Drop exemplare;
drop titluri_autori;
drop titluri_cuvinte;

```
create table exemplare (
idcota number(15) not null primary key
,formatul varchar2(15)
,nrpag number(4)
,idisbn number(13) not null references titluri(idisbn)
)
/

create table titluri_autori(
idisbn number(13) not null references titluri(idisbn)
,autor varchar2(30) not null unique
,primary key(idisbn,autor)
)
/
```

 ISSN 2069-7988
ISSN-L 2069-7988

```
create table titluri_cuvinte(
idisbn number(13) not null references titluri(idisbn)
,cuvinte varchar2(30)
,primary key(idisbn,cuvinte)
)
```

Dictionarul de date pastreaza informatii despre restrictii intr-o tabela virtuala speciala numita USER_CONSTRAINTS a carei structura este vizualizata in sql*plus
Select constraint_name,table_name,position
From user_cons_columns
Where table_name=’exemplare’;
Pentru atribuirea unei nume explicit pentru fiecare restrictie se utilizeaza clauza CONSTRAINT
Pentru usurarea lucrului cu baze de date se foloseste notatia pseudo, anume

- Pk_(primary key)pentru cheile primare;
- Un_(unique) pentru cheile alternative;
- Nn_(not null) pentru atributele obligatorii (ce nu pot avea valori nule);
- Ck_(check)pentru reguli de validare la nivel de atribut sau inregistrare;
- Fk_(foreign key)pentru cheile straine.

Exemplu 26.Sa se creeze tabela exemplare folosind clauza CONSTRAINT

```
create table exemplare (
idcota number(15)
CONSTRAINT NN_EXEMPLARE _idcota not null
CONSTRAINT pk_exemplare_idcota primary key
,formatul varchar2(15)
,nrpag number(4)
,idisbn number(13)
CONSTRAINT nn_exemplare_idisbn not null
CONSTRAINT fk_exemplare_idisbn references titluri(idisbn)
)
/
```

Pentru a afla informatii despre restrictiile definite pe fiecare tabela din dictionarul de date se utilizeaza clauza USER_CONS_COLUMNS
Exemplu 27 SELECT constraint_name,column_name,position
From user_cons_columns
Where table_name=’Exemplare’;
Exemplu 28.Daca dorim sa aflam componenta cheii primare a tabelei exemplare

Select constraint_name,column_name,position
From user_cons_columns
Where constraint_name=’pk_exemplare’;
observatie
La incalcarea restrictiilor de cheie primara si de unicitate, se declanseaza o eroare
La declararea restrictiilor de tip cheie primara/unicitate, Oracle creeaza automat indecsi corespunzatori.

6.Reguli de validare la nivel de atribut si inregistrare cluaza CHECK

Restricţiile utilizator , denumite restricţii de comportament sunt implementate sub forma regulilor de validare la nivel de câmp (field validation rule), la nivel de inregistrare(record validation rule) sau, dacă sunt complexe, sunt incluse în declanşatoare(triggere)

Pentru tabela personal?biblioteca avem următoarele restricţii

1.atributul Marca nu poate avea valori mai mici de 100

2.pentru nume şi prenume, prima literă trebuie să fie majusculă, restul litere mici.

Exemplu 29.Sa se introduca clauza CHECK

```
Drop table personal_biblioteca ;
Create table personal_bioblioteca(
Marca integer
,constraint pk_personal_biblioteca primary key
,constraint nn_personal_biblioteca_marca NOT NULL
,constraint ck_personal_biblioteca_marca CHECK(marca>==100)
,numepren varchar2(35)
,constraint nn_personal_biblioteca_numepren not null
,constraint ck_personal_biblioteca_numepren
CHECK(numepren=ltrim(initcap(numepren)))
……..

) ;
```

Exemplu 30 Sa se creeze tabela titluri_autori de carti din cadrul SGBDD Biblioteca

```
CREATE TABLE titluri_autori(
Isbn char(13)
Constraint fk_titluri_autori_isbn references titluri(isbn)
,autor varchar2(30)
Constraint ck_titluri_autori_autor check(autor=ltrim(initcap(autor)))
,constraint pk_titluri_autori primary key(isbn,autor)
)
/
```

Exemplu 31.Pentru aflarea regulilor de validare la nivel e camp si inregistrare din tabela titluri_autori se utlizeaza consultarea.

```
SELECT constraint_name,table_name,search_condition
From user_constraints
Where table_name IN(TITLURI_AUTORI')
AND CONSTRAINT_TYPE='c'
```

7.Restrictii referntiale

SGBD Oracle utilizeaza clauza REFERENCES/FOREIGN

EXEMPLU 32

```
CREATE TABLE titluri_autori(
```

Isbn char(13)
Constraint fk_titluri_autori_isbn references titluri(isbn)
,autor varchar2(30)
Constraint ck_titluri_autori_autor check(autor=ltrim(initcap(autor)))
,constraint pk_titluri_autori primary key(isbn,autor)
,……
Regula de stergere in cascada (ON DELETE CASCADE) la stergerea unui parinte se sterg automat toti copiii sai :

Exemplu 33

CREATE TABLE titluri_autori(
Isbn char(13)
Constraint fk_titluri_autori_isbn references titluri(isbn)
ON DELETE CASCADE
,autor varchar2(30)
Constraint ck_titluri_autori_autor check(autor=ltrim(initcap(autor)))
,constraint pk_titluri_autori primary key(isbn,autor)
,……
Pentru a nu incalca restrictia referentiala la stergerea unei inregistrari parinte toate valorile copil pot fi trecute pe NULL (ON DELETE SET NULL)
Create table titluri_autori(
Isbn…….
,constraint fk_titluri_autori_isbn references titluri(isbn)
On delete set null
…….
Cea mai rezonabila optiune este cea implicita, care interzice stergerea unei inregistrari parinte, atat timp cat exista macar un copil
On delete restrict
Exemplu 34 Pentru afla informatiile despre resrictiile referentiale acestea sunt pastrate in dictionarul de date
Select constraint_name,table_name,r_constraint_name
From user_constraints
Where constraint_type=’R’;

Exemplu 35.Extragerea corespondentelor dintre atributele copil si cele parinte este o sarcina mai dificila si presupune executia unei fraze SELECT mul mai elaborate.

SQL>select uc.constraint AS restrictie_copil,uccl.table_name AS tabela_copil,
R_constraint_name AS restrictie_parinte,
Ucc2.table_name as tabela_parinte,
Ucc2.column_name as atribut_parinte
FROM user_constraint uc,user_cons_columns ucc1,
User_cons_columns ucc2
Where uc.constraint_name=ucc1.constraint_name an
duc.r_constraint_name=ucc2.constraint_name and ucc1.position=ucc.position an
duc.constraint_type=’R’
Order by uc.constraint_name,ucc1.position

Exemplu 36 Sciptul de creare a tabelelor si declararea restrictiilr

```
drop table imprumut;
drop table clienti;
drop table titluri_cuvinte;
drop table titluri_autori;
drop table exemplare;
drop table titluri;
drop table edituri;
drop table czu;
drop table autori;
drop table reviste;
drop table ziare;
drop table carti;

create table autori(
grupa varchar2(9) primary key
,cifra number(2)
)
/


create table czu(
dendiviz varchar2(35) primary key
,clasa char(15)
)
/

create table edituri(
editura varchar2(30) primary key
,locsediu varchar2(30)
,adresa  varchar2(30)
,telefon char(10)
,mobil char(10)
,email varchar2(30)
,wwww varchar2(30)
)
/

create table titluri(
idisbn number(13) not null primary key
,titlu varchar2(60) not null
,editura varchar2(30) not null references edituri(editura)
,nrinv number(6)
,nrfact number(15)
```

```
,anaparitie number(4) default extract (year from current_date)
,dataprim date default sysdate
,nrexem number(4)
,pret number(12,2)
,repartiz varchar2(20)
,mentiuni varchar2(20)
)
/

create table exemplare (
idcota number(15) not null primary key
,formatul varchar2(15)
,nrpag number(4)
,idisbn number(13) not null references titluri(idisbn)
)
/

create table titluri_autori(
idisbn number(13) not null references titluri(idisbn)
,autor varchar2(30) not null
,primary key(idisbn,autor)
)
/

create table titluri_cuvinte(
idisbn number(13) not null references titluri(idisbn)
,cuvinte varchar2(30)
,primary key(idisbn,cuvinte)
)
/

create table clienti(
nrpermis number(6) not null primary key
,nume varchar2(25)
,pren varchar2(15)
,adresa varchar2(30)
,telefon char(10)
,loc varchar2(15)
,telem char(10)
,bi char(15)
,eliberat varchar2(15)
,datae date default sysdate
,datanast date default sysdate
,studii varchar2(15)
,ocupatia varchar2(25)
)
/
```

```
create table imprumut(
nrpermis number(6)
,idisbn number(13)
,isbn varchar2(13)
,nrinv number(6)
,dataimp date default sysdate
,autor varchar2(30)
,titlu varchar2(60)
,czu varchar2(20)
,cota varchar2(15)
,numepren varchar2(30)
,datarest date default sysdate
)
/
```

9.Modificarea structurii tabelelor :comanda ALTER TABLE
9.1 Adaugari/Modificari/Stergeri de atribute
Adaugarea atributului nrpag (nr. De pagini)in tabela titluri_autori se realizeaza astfel:

```
ALTER TABLE Titluri_autori ADD nrpag number(4);
```

stergerea unui atribut din tabela titluri nr.exemplare
Exemplu 37

```
ALTER TABLE Titluri drop column nr.exemplare;
```

exemplu 38. Stergerea unui articol la nivel de inregistrare se face astfel:

```
sql>ALTER TABLE titluri DROP COLUMN ISBN CASCADE CONSTRAINTS;
```

stergerea definitiva a tuturor atributelor marcate ca inutilizabile se realizeaza prin :

```
ALTER TABLE titluri DROP UNUSED COLUMNS
```

EXEMPLU 39 Vizualizarea informatiilor despre campurile eliminate

```
SQL>SELECT * FROM USER_UNSED_COL_TABS;
```

EXEMPLU 40 Modificarea unui camp

```
ALTER TABLE Titluri Modify EDITURA VARCHAR2(30)
/
```

MESAJ DE EROARE
ORA=1441:cannot decrese column length because some value is too big

In Oracle modificarea unui atribut se poate face astfel; din CHAR in VARCHAR2 sau VARCHAR
Si invers

```
ALTER TABLE Tiluri Modify editura char(20);
```

9.2Adugari/stergeri de restrictii
Interzicera valorilor nule pentru atributul isbn se realizeaza astfel

```
SQL>ALTER TABLE Tiluri MODIFY isbn NOT NULL;
```

si invers
Toate restrictiile, adica valori nenule, cheie primara, unicitate, reguli de validare,restrictii referentiale pot fi declarate su ulterior crearii tabelelor si sterse la un moment dat.

Exemplu 40.Sa se construiasca o restrictie astfel:

ALTER TABLE EDITURI CONSTRAINT ck_edituri_editura
(check(editura=ltrim(initcap(editura)))

Exemplu 41 Stergerea cheie primare editura nu se poate face direct

ALTER TABLE EDITURI DROP PRIMARY KEY
Mesaj de eroare ORA-02273.....
Metoda este :
SQL>ALTER TABLE edituri drop constraint pk_edituri;
sau
ALTER TABLE titluri DROP PRIMARY KEY CASCADE;

10.Restrictii dezactivate,reactivate, amanate, imediate
1.Dezactivarea restrictiei CHECK :ck_edituri_editura asociata tabelului Edituri se realizeaza folosind comanda ALTER TABLE in formatul:

ALTER TABLE EDITURI DISABLE CONSTRAINT ck_edituri;
reactivarea presupune executia comenzii:
ALTER TABLE EDITURI ENABLE CONSTRAINT ck_edituri1;
Stergerea de atribute si restrictii, dezactivarea simpla este :
Exemplu 41
ALTER TABLE EDITURI DISABLE CONSTRAINT pk_edituri;
se soldeaza cu un mesaj de eroare (ORA-02297.....), iar solutia tine tot de cluaza cascade

Exemplu 42 Sa se adauge un camp ore in tabela PERSONAL
Dupa care sa modifice tabela pentru ore-lucrate 10 h
Algoritmul este:
ALTER TABLE PERSONAL ADD CONSTRAINT ck_personal_ore
Check(ore BETWEEN 0 AND 10) ENABLE NOVALIDATE;
EXEMPLU 43 Sa se declare o integritate referentiala a titluri ca amanabila
ALTER TABLE titluri CONSTRAINT fk_edituri_titluri;
alter table add
  constraint fk_edituri_titluri foreign key(isbn)
  refrences titluri(isbn)
deferable initialiy immediate;
restrictia fk_edituri_tiluri va fi amanabila, insa clauza INITIALIY IMMEDIATE determina deocamdata verificarea acesteia la fiecare comanda SQL
Trecerea din starea amanabila imediata in amanabila amanata se face folosind comanda
SET CONSTRAINT fk_edituri_titluri DEFERED;

11 Scipturi SQL*PLUS de refacere a tabelelor si restrictiilor
In cazul cand dorim mutarea bazei de date pe alt server, fara a recurge la operatiunile de export/import, cat si salvarea structurii bazei de date se poate re-crearea tabelelor folosind comenzi de lucru in Oracle

11.1 Re-crearea tabelelor

Daca numele tabelelor pot fi preluate din tabela virtuala a catalogului sistem USER-TABLES , numele, tipul, lungimea si valorile implicite ale atributelor se regasesc in USER_TAB_COLUMNS
Scriptul se obtine in SQL*Plus prin comanda SPOOL,prin care rezultatele frazei SELECT urmatoare vor fi salvate in fisierul ASCII re_creare_tabele.sql din directorul e:\biblio\migrare_oracle\creare_tabele.sql
Exemplu 43

```
Column "—Creare_tabele "FORMAT A60
Column C2 FORMAT A40

Spool c:\biblio\migrare_oracle\re_creare_tabele.sql
SELECT
'CREATE TABLE '||table_name||'('AS"—Creare_tabele",
'-'||RPAD(table_name,30)||'' as c2
FROM user_tables
UNION
SELECT CASE column_id WHEN 1 THEN '' ELSE ',' END ||
      Column_name||''||data_type||
       CASE WHEN data_type NOT IN ('NUMBER','CHAR','VARCHAR2')
THEN ''
               ELSE '('||
                 NVL(data_precision,data_length)||
                    CASE WHEN data_scale >0 THEN','|| data_scale ELSE " END
     ||')'
'-'||RPAD(table_name,30)||TO_CHAR(column_id,'999')
FROM user_tab_columns
UNION
SELECT ') :'||CHR(13).'-'||RPAD(table_name,30)||''||CHR(123)
FROM user_tables
UNION
Select ''||chr(13)'-'||RPDA(table_name,30)||''||CHR(124)
FROM user_tables
ORDER BY 2;


Spool off
```

Cele trei parti ale comenzii CREATE TABLE:

- **create table tabela (**
- **atribut1 tip (lungime[,poz_fractionara1]) [,atribut2 tip(lungime,poz_fractionara2[)**
- **);**

sunt generate de trei fraze SELECT , conectate prin operatorul UNION
executia sriptului se face astfel :

```
SQL>conn biblio/biblio
Sql>@c:\biblio\migrare_oracle_re_creare_tabele.sql;
```

 ISSN 2069-7988
ISSN-L 2069-7988

112. Re-crearea clauzelor DEFAULT
PENTRU A SALVA SITUATIA DE RE-CREAREA A CLAUZELOR default S-A FOLOSIT O TABELA TEMPORARA temp1, IN CARE ATRIBUTUL VAL_IMPLICITA ESTE DE TIP clob, iar la inserarea liniilor din USER_TAB_COLUMNS s-a folosit functia TO_LOB
Exemplu 44 Extragerea valorilor implicite

```
DROP TABLE temp1;
CREATE TABLE temp1(
Tabela VARCHAR2(30),
atribut VARCHAR2(30),
id_atribut NUMBER(2),
tip_atribut VARCHAR2(20),
val_implicita CLOB
);


Insert into TEMP1
Select table_name,column_name,column_id,data_type, TO_LOB(data_default)
FROM user_tab_columns
WHERE data_default IS NOT NULL;


COMMIT;

spool c:\biblio\migrare_oracle\re_creare_default.sql

SELECT 'ALTER TABLE '||tabela||'MODIFY'||atribut||'DEFAULT'
||val_implicita||';' AS ""
FROM temp1;

spool off
```

Fisierul ASCII generat prin comanda spool este re_creare_default.sql

11.3 Restrictii NOT NULL
Pentru fiecare atribut declarat NOT NULL, cream o restrictie cu numele nn_tabela_atribut

Exemplu 45 Script pentru re-crearea_notnull.sql

```
Spool c:\biblio\migrare_oracle\re_creare_notnull.sql

SELECT'ALTER TABLE '||table_name||'MODIFY('||column_name
||'CONSTRAINT NN_'||table_name||''||column_name||'NOT NULL);'
FROM user_tab_columns
WHERE nullable='N'
```

```
ORDER BY table_name,column_name;

spool off
```

11.4 Chei primare si alternative
Cheile primare si alternative le corespund in tabela virtuala din dictionar USER_CONSTRAINTS
Cate o restrictie de tip ‘P’ si ‘U’
Exemplu 46 Script de re-crearea cheilor primare si alternative

```
Spool c:\biblio\migrare_oracle\re_creare_chei_primare_si_alternative.sql

SELECT ‘ALTER TABLE ‘||table_name||’DROP CONSTRAINT’||
Constraint_name||’CASCADE;’as”—Stergere restrictie”
From user_constraints
Where CONSTRAINT_TYPE in(‘P’,’U’);

SELECT ‘ALTER TABLE’||table_name||’ ADD CONSTRAINT pk_’||table_name||
‘PRIMARY KEY’||’(‘||atrr1||
CASE WHEN atrr2 IS NOT NULL THEN ‘,’||atr2 END||
CASE WHEN atr3 IS NOT NULL THEN’,’||atr3 END
||’);’ as”—Cheile primare”
From user_constraints uc
Left outer join
(select constraint_name,column_nama as atr1
From user_cons_columns
Where position=1)a1 on uc.constraint_name=a1.constraint_name
Left outer join
(select constraint_name,column_name as atr2
From user_cons_columns
Where position=2)a2 on uc.constraint_name=a2.constraint_name
Left outer join
select constraint_name,column_name as atr3
From user_cons_columns
Where position=3)a3 on uc.constraint_name=a3.constraint_name
Where constraint_type=’P’;


Select ‘alter table’||table_name||’ADD CONSTRAINT un_’||table_name||
‘-‘||atr1||

CASE WHEN atrr2 IS NOT NULL THEN ‘-‘||atr2 END||
CASE WHEN atr3 IS NOT NULL THEN’-’||atr3 END
||’UNIQUE’||’(‘atr1||
CASE WHEN atrr2 IS NOT NULL THEN ‘,’||atr2 END||
CASE WHEN atr3 IS NOT NULL THEN’,’||atr3 END
||’);’ as”—Cheile aleternative”
```

```
From user_constraints uc
Left outer join
(select constraint_name,column_nama as atr1
From user_cons_columns
Where position=1)a1 on uc.constraint_name=a1.constraint_name
Left outer join
(select constraint_name,column_name as atr2
From user_cons_columns
Where position=2)a2 on uc.constraint_name=a2.constraint_name
Left outer join
select constraint_name,column_name as atr3
From user_cons_columns
Where position=3)a3 on uc.constraint_name=a3.constraint_name
Where constraint_type='U';
spool off
```

Structurile de tip CASE sunt limitate la trei coloane atr1,atr2, atr3

11.5 Reguli de validare la nivel de atribut si inregistrare

Pentru a extrage informatii despre regulile de validare trebuie sa :

- Dictionarul de date le stocheaza in tabele USER_CONSTRAINTS cu valoarea 'C' pentru atributul constraint_type
- Expresia ce formeaza restrictia reprezinta valoarea coloanei search_condition care este de tip LONG
- Oracle include si pe cele de tip NOT NULL

Pentru aceasta folosim tabela temp1 pentru conversia atributului LONG in CLOB

EXEMPLU 47 Script pentru re-crearea regulilor de validare

```
Spool c:\biblio\migrare_orace\re_creare_check.sql
Drop table temp1;
create table temp1 (
restrictie VARCHAR2(30),
tip_restrictie varchar2(20),
tabela varchar2(30),
expresie CLOB
);
insert into temp1
SELECT constraint_name,constraint_type,table_name,
to_lob(search_condition)
from user_constraints
where constraint_type='C';

COMMIT;

select 'alter table'||TABELA||'drop constraint'||RESTRICTIE||';'
as""—Stergere reguli de validare"
from temp1
where expresie NOT LIKE '%NULL%';

SELECT 'ALTER TABLE'||tabela||' ADD CONSTRAINT'||restrictie||
```

```
'CHECK('||CAST(expresie AS VARCHAR2(300)||');' as "—CHECK-URI""
FROM TEMP1
WHERE expresie NOT LIKE '%NULL%';

spool off
```

11.6 Restrctiile referentiale
Avem de identificat componenta cheilor straine din tabela copil, ci si pe cea a cheilor corespondente din tabela copil
Dictionarul Oracle ofera tabela virtuala USER_CONSTRAINTS, furnizeaza informatii despre numele restrictiei din tabela copil (constraint_name) pe baza careia pot fi identificate cheile straine, cat si numele restrictiei primare/unice corespunzator din tabela parinte 9r_constraint_name)

Exemplu 48 Script de re-crearea-restrtiile-referentiale

```
Spool c:\biblio\migrare_oracle\re_creare_restrictii_referentiale.sql

SELECT'ALTER TABLE'||table_name||'DROP CONSTRAINT'||
Constraint_name||'CASCADE;' as "—Stergere restr.refrentiale"
From user_constraints
Where constraint_type='R';

select'alter table '||uc_copil.table_name||'ADD CONSTRAINT '||constraint_name||
'FOREIGN KEY'||'('||strc1|\)
Case when atrc2 is not null then ','||atrc2 end||
Case when atrc3 is not null then ','||atrc3 end
||')'||
'refrences'||uc_parinte.table_name||'('||atrp1||
Case when atrp2 is not null then ','||atrp2 end||
Case when atrp3 is not null then ','||atrp3 end
||')' as "—restrictii refrentiale"
From user_constraints uc_copil
INNER JOIN user_constraint uc_parinte
On uc_copil.r_constraint_name=uc_parinte.constraint_name
Left outher join
(select  constraint_name,column_name as atrc1
From user_cons_columns
Where position=1)c1 on uc_copil.constraint_name=c1.constraint_name
Left outher join
(select  constraint_name,column_name as atrc2
From user_cons_columns
Where position=2)c2 on uc_copil.constraint_name=c2.constraint_name
Left outher join
(select  constraint_name,column_name as atrc3
From user_cons_columns
Where position=3)c3 on uc_copil.constraint_name=c3.constraint_name
```

```
Left outher join
(select  constraint_name,column_name as atrp1
From user_cons_columns
Where position=1)p1 on uc_parinte.constraint_name=p1.constraint_name

Left outher join
(select  constraint_name,column_name as atrp2
From user_cons_columns
Where position=2)p2 on uc_parinte.constraint_name=p2.constraint_name

Left outher join
(select  constraint_name,column_name as atrp3
From user_cons_columns
Where position=3)p3 on uc_parinte.constraint_name=p3.constraint_name

WHERE uc_copil.constraint_type='R';

spool off
```

Exemplu 50 Sa se creeze urmatoarele tabele din baza de date Biblioteca, folosind scriptul general
Exemplu .Modificarea structurii unei tabele presupune adaugarea de noi coloane la sfarsitul acesteia sau modificarea tipurilor unor coloane deja existente
Comanda care realizeaza aceste operatii este ALTER TABLE. Sintaxa ei este :
ALTER TABLE nume-tabela[ADD!MODIFY}(spec_col [NULL!NOT NULL,.....])
Unde :
ADD-adauga la sfarsitul tabelei noi coloane care initial au valori nule, ulterior prin actualizare
Modify – schimba definirea unei coloane existente
Null-sa refera la faptul ca valorile din coloana ce sa adauga /modifica sunt nule
Not null- specifica faptul ca o coloana nu poate avea valori nule
Exemplu 3 Sa se adauge la tabela autori o noua coloana grupa1 de c(9)

```
SQL>ALTER TABLE AUTORI ADD
      2 (grupa1 char(9));
```

mesaj aparut pe ecran :table altered
exemple 10
Stergerea unei tabele dintr-o baza de date biblioteca se utilizeaza comanda DROP TABLE
Sintaxa ei este :
DROP TABLE nume_tabela;
exemplu 51. Sa se sterga tabela carti

```
sql>drop table carti;
```

2.5.5 Crearea si stergerea sinonimelor pentru tabele sau viziuni

Pentru a simplifica accesul la tabele sau la viziuni se pot crea sinonime
Sintaxa generala este :CREATE SYNONYM
SQL>CREATE [PUBLIC] SYNONYM nume FOR [nume-utilizator]
Nume_tabela [@baza_de_date];
unde :
public permite crearea unui sinonim public la care orice utilizator se poate referi fara a fi necesara calificarea.
Exemplu 52 Sa se creeze carti1 pentru tabela carti din baza de date Biblio, prin utilizatorul util1
Sql>CREATE SYNONYM CARTI1
2 FOR UTIL1.CARTI @BIBLIO;
Stergerea unui sinonim este urmatoarea :DROP [PUBLIC] SYNONYM nume-sinonim;
2.5.6 Crearea si stergerea unei viziuni
Crearea unei viziuni se realizeaza cu comanda CREATE VIEW
Sintaxa generala este :
CREATE VIEW nume_viziune
[(numecol,nume_col.....)] As cerere
[WITH CHECK OPTION];
EXEMPLU 53 Sa se creeze vizuunea cu numele vcarti folosind tabela carti;
SQL>CREATE VIEW VCARTI AS SELECT *
2 FROM CARTI
3 WHERE Editura='All';
Exemple 5
2.1.7 crearea ,validarea si stergerea indecsilor pentru o tabela In vederea obtinerii de perfomante privind accesul la tabelele unei baze de date se pot construi indecsi folosind comanda CREATE INDEX
SQL>CREATE INDEX [UNIQUE] nume_index on nume_tabela (nume_col [ASC!DESC].....
[COMPRESS!NOCOMPRESS]
[SYSSORT!NONSYSORT]
[ROWS=n][PCTFREE=[20!n]];
Unde :Unique=se refera la faptul ca nu va contine doua randuri cu aceleasi valori
Compress:indica faptul ca indecsii pot fi comprimati
Pctfree-indica in momentul crearii indexului procentul din spatiul pentru index ce trebuie sa ramana liber pentru actualizari.
Comanda pentru validare a unui index este :VALIDATE INDEX
Sintaxa generala este :
VALIDATE INDEX nume_index [ON nume-tabela] [WITH LIST];
unde :
with list-salveaza informatiile necesare la index intr-un fisier

Exemplu 58 Sa se creeze un index cu numele indexcarti pentru coloana editura din tabela carti
SQL>CREATE INDEX INDEXCARTI ON CARTI(EDITURA);
EXEMPLU 59.Tipuri de indecsi
a)index de tiop arbore B (B-tree index sau balanced tree index)

un arbore de tip B este un arbore in care pentru gasirea oricarei valori din arbore sunt necseari acelasi numar de pasi
exemplu de index de tip arbore B
schema de 6.5
b)index partitionat pentru a crea un index partitionat se foloseste comanda CREATYE INDEX cu clauza PARTITION
exista doua moduri de a defini un index partitionat :local si global

- Index local sunt similare cu partitiile tabelului, in acest fel indexul este partitionat in functie de aceeasi coloana
- Global partitiile indexului sunt definite de utilizator si nu sunt similare cu partitiile tabelului la care se refera indexul.

Sintaxa pentru indexul local este :
CREATE INDEX nume_tabela
On tabel (coloana[,coloana].....)
Local
(partition nume_partitie [pctfree intreg][pctfree intreg][tablespace spatiu tabel][storage parametrii de stocare].....]
Exemplu 60 Sa se creeze un index partitionat local pe baza tabelului partitionat carti_partitiont
SQL>CREATE INDEX nume_carti_ind
On carti_partitonat(editura)
Local
(partition carti_scurte tablespace ts_ind_alfa storage (initial 10 k next 10k),
partition carti_mediu
tablespace ts_ind_beta
storage (initial 20k next 20k)
);
crearea unui index global
create index nume_index
on tabel (coloana[,coloana].....)
Global
Partition by range (lista_coloane)
Partition nume_partitie values [less!granter]tha (;ista_valori)
[pctfree intreg][pctfree intreg][tablespace spatiu_tabel][storage parametri_de_stocare].....
Exemplu 61 Sa se creeze indexul global nume_carti_ind pentru tabela carti,indexul va avea doua partitii
SQL>CREATE INDEX nume_sal_ind
ON carti(editura)
Global
Partition by range(nume)
(partition values less than (‘n’)
Tablespace ts_alfa_ind,
Partition values less than (maxvalue)
Tablespace ts_beta_ind);

c)index de cluster (index de grup) este un index bazat pe coloanele commune ale unui cluster.Nu se pot executa nici un fel de comenzi DML asupra unui cluster pana cand nu a fost create un index de cluster
d)index cu cheie inverse
un index cu cheie inverse se creaza folosind comanda CREATE INDEX cu optiunea REVERSE
exemplu 62 CREATE INDEX sal_preume_ind ON salariat(prenume)REVERSE;
Un index obisnuit se poate transforma in index cu cheie inversa folosind co manda ALTER INDEX.....REBUILD cu optiunea REVERSE
EXEMPLU 63.ALTER INDEX sal_prenume_ind REBUILD REVERSE;
e)index de tip Bitmap
bitul este 1 daca valoarea respective este continuta in acel rand si 0 daca nu este
exemplu 64 Fie tabelul masina

Nr_masina	Marca	culoare
1	Dacia nova	Alb
2	Logan	Rosie
3	For mondeo	Neagra
4	Dacia supernova	verde

Index de tip bitmap pentru coloana culoare

Alb	Rosie	Neagra	Verde
1	0	0	0
0	1	0	0
0	0	1	0

Exemplu 64 Pentru a afla numarul de masini albe se va executa interogarea:
SQL>SELECT COUNT(*) FROM masina where culoare='alb';
Crearea unui index de tip bitmap se face folosind comanda CREATE INDEX cu optiunea BITMAP
Exemplu 65 SQL>CREATE BITMAP INDEX culoare_ind ON masina(culoare);
Observatie
Spre deosebire de indecsii traditionali de tip arbore B, folosirea indecsilor de tip index bitmap se recomanda atunci cand:
-numarul de valori distincte ale coloanei indexate este relative mic(coloana are cardinalitatea mica) exemplu :starea civila a unei personae
-majoritatea interogarilor contin combinatii de conditii WHERE ce implica operatorul OR
- 2.5.8 Tabele organizate pe baza de index
Exemplu tabelul cu diferentele cele mai importante dintre un tabel obisnuit si un tabel organizat pe baza de index

Tabel obisnuit	Tabel organizat pe baza de index
ROWID identifica in mod unic un rand Cheia primara este optionala	Cheia primara indentifica in mod unic un rand, specificarea cheii primare este obligatorie
Are coloana implicita ROWID	Nu are coloana implicita ROWID

2.5.9 Crearea tabelelor organizate pe baza de index
Pentru a crea un tabel , se foloseste comanda SQL CREATE TABLE cu specificatia ORGANIZATION INDEX.La crearea unui tabel organizat pe baza de index, trebuie

specificata cheia primara, respectiv clauzele OVERLOW, THRESHOLD, INCLUDING
EXEMPLU 66 Sa se creeze tabel organizat pe index carte
Sql>CREATE TABLE CARTE(
Serie varchar2(5),
numarfactura number(7),
titlu varchar2(40),
cod_autor varchar2(10),
editura varchar2(20),
descriere varchar2(200),
constraint pk_carte primary key(serioe,numarfactura))
organization indexed
tablespace ts_alfa_carte
pctthreshold 40
including editura
overlow tablespace ts_beta_carte;
2.5.10 Clustere
a)clusterul de index (grupul de tabele) reprezinta o metoda optionala de stocare a datelor
cheia unui cluster este definita ca fiind coloana sau coloanele pe care tabele grupate le au in comun
Crearea unui tabel intr-un cluster de index
Pentru a crea un cluster de index se parcurg urmatoarele etape :
1.crearea clusterului de index
2.adaugarea la cluster
3.crearea indexului de cluster
Adaugarea la cluster se poate face si dupa ce indexul de cluster a fost creat
Sintaxa generala pentru crearea clusterului de index este :
CREATE CLUSTER nume_cluster (nume_col tip data[,.....)
[PCTFREE intreg][Pctfree intreg][size intreg][tablespace spatiu_tabel][storage parametrii_de_stocare]
Exemplu 67 Sa se creeze un cluster sal_dep in care tabele sunt grupate dupa coloanele cod_dep si cod_tara
SQL>CREATE CLUSTER sal_dept
(cod_dept numer(10),
Cod_tara nmber(10));
2.adaugarea tabelelor la cluster
Comanda cu care se creaza este :CREATE TABLE cu clauza CLUSTER
SQL>CREATE TABLE nume_cluster
(nume_coloana tip data.....[,.....)
CLUSTER nume_cluster
Exemplu 68 Sa se creeze tabelul departament cu cluaza cluste sal_dep
SQL>CREATE TABLE departament(
Cod_dept number(10),
cod_tara number(10),
nume_dep varchar2(35),
PRIMARY KEY(cod_dept,cod_tara))
CLUSTER sal_dept9cod_dept,cod_tara);

Exemplu 69 Sa se creeze tabele titluri folosind tabela parinte edituri cu cheia striana editura
3.index la cluster
Pentru coloanele cheie ale clusterului creat in mod explicit un index de cluster
Pentru crearea unui index de cluster se foloaseste comanda SQL.CREATE INDEX cu optiunea ON CLUSTER
EXEMPLU 70 Sa se creeze indexul pentru cluster sal_dept;
SQL>CREATE INDEX sal_dept
ON CLUSTER sal_dept ;
2.5.11 clusterul hash
Acest cluster de hash foloseste o functie hash
Crearea unui table intr-un cluster hash se face astfel:
1.crearea clusterului hash;
2.adaugarea tabelului la cluster.

1.crearea clusterului hash
Pentru crearea clusterelor se utilizeaza comanda sql. CREATE CLUSTER cu optiunea HASHKEYS
Sintaxa generala este urmatoarea :
CREATE CLUSTER nume_cluter
(nume_coloana tip_data[,.....))
HASHKEYS intreg
[hash is expresire]
[size intreg]
[pctfree intreg][pctused intreg]
[tablespace spatiu_tabel]
[storage parametrii_de _stocare]
Exemplu 71 SQL>CREATE CLUSTER test_cls(cod numaer(5))
HASHKEYS 1000
HASH IS COD
SIZE 500;
2 adaugarea tabelului la cluster se face folosind CREATE TABLE cu clauza CLUTER
EXEMPLU 72 SQL>CREATE TABLE test
(cod number(5) primary key,
…….)
Cluster test_cls(cod);
3.Modificarea clusterelor se face folosind comanda ALTER CLUSTER
4.Stergerea clustrelor
Exemplu 73 Sa seterga clusterul sal_dept
SQL>DROP CLUSTER sal_dept
INCLUDING TABLES;
EXEMPLU 74 Daca exista tabele din afara clusterului care contin restrictii de integritate care se refera la chei din tabele clusterului acestea trebuie eliminate, pentru acest lucru se foloseste comanda CASCADE CONSTRAINTS
SQL>DROP CLUSTER sal_dept
INCLUDING TABLES CASCADE CONSTRAINTS;

Capitolul 3 Actualizarea datelor

In functie de momentul in care se doreste realizarea actualizarilor asupra bazei de date, utilizatorul poate folosi una din urmatoarele comenzi:
a)SET AUTOCOMMIT IMM-schimbarile se efectueaza imediat;
b)SET AUTOCOMMIT OFF-schimbarile sunt pastrate intr-un buffer pana la executia uneia din comenzile COMMIT WORK care are rolul de a permanentiza schimbarile efectuate; ROLLBACK WORK care determina renutarea schimbarilor realizate.

3.1 Adaugarea randurilor intr-o tabela

Se utilizeaza comanda INSERT cu forma generala :
INSERT INTO nume_tabela
[(nume_col1,nume_col2,.....)]
{VALUES (valoare1,valoare2,.....};
Exemplu 80 Sa se adauge in baza de date edituri
SQL>INSERT INTO EDITURI VALUES('All','Bucuresti');

3.2 Inserari si secvente

Secventa este un obiect al unei baze de date care genereaza valori unice
In cazul aplicatiilor cu mai multi utilizatori orice atribut este o cheie surogat
Sintaxa generala este :
CREATE SEQUENCE seq_camp
INCREMENT BY 1
MINIVALUE 100 MAXVALUE 999999999
NOCYCLE NOCACHE ORDER;
numele secvente este seq_camp
valorile generate vor fi consecutive increment by 1 incapand de la 100 pana la val.maxima 9999999999
clauza nocycle la urmatorul apel al unei valori din secventa va fi generata o eroare (ORA-08004)
EXEMPLU 81 Sa se creeze secventa edituri
SQL>CREATE SEQUENCE seq_idisbn
Increment by 1
Minivalue 1 maxvalue 99999999999
Nocycle nocache order;
dupa crearea secventei o valoare va fi generata folosind clauza NEXTVAL;
valoarea curenta (ultima valoare generata) a secventei poate fi obtinuta prin clauza CURVAL
EXEMPLU 82 Sa se genereze valorile din secventa
SQL>SELECT seq_idisbn.NEXTVAL FROM DUAL;
SQL>SELECT seq_idisbn.CURVAL FROM dual;
Atentie !
Clauza CURVAL nu poate fi folosita decat daca secventa a fost initializata prin NEXTVAL, altfel va apare o eoare ORA-08002.....
EXEMPLU 83
SQL>DISCONNECT
SQL>CONN bibli/biblio
Connected.
SQL>
SQL>SELECT seq_idisbn.CURVAL FROM dual;

SELECT seq_idisbn.CURVAL FROM dual
Sql>
Pentru aflarea valorii curente din secventa fara a recurge la NEXTVAL se poate face folosind o comanda din dictionarul de date USER_SEQUENCES
SQL>DESC user_sequences;
sql>select * from user_sequences;
Exemplu 84 Sa se introduca date in baza de date edituri folosind secventa
SQL>Scrip de populare a tebelei Edituri

Delete from edituri;
drop sequence seq_idisbn;
create sequence seq_idisbn increment by 1
minivalue 1 maxvalue 99999999999 nocycle nocache order;
insert into edituri values(seq_idisbn.nextval,'All','Bucuresti');
insert into editiuri values(seq_idisbn.nextval,'Polirom',;'Iasi');
COMMIT;

3.3 Inserare prin subconsultare

Exemplu 85
INSERT INTO titluri (isbn,tilu,editura,anaparite)
Select editura,locsediu from edituri;
3.4 inserari in tabele multiple
Exemplu 86 Scipt de crearea tabelelor de arhivare
Create table titluri_ian as select * from titluri where 1=2;
.....
Create table titluri_decm as select * from titluri where 1=2;
3.4 stergerea liniilor –delete
Exemplu 86 sa se sterga liniile tabelului edituri;
delete from edituri where editura='All';
exemplu 87
DELETE FROM titluri where(isbn,editura)in
(select isbn,editura from titluri_ian);
3.5 Modificarea valorilor UPDATE
Exemplu 88
Update personal_biblioteca set salariu=salariu*1.10;
exemplu 89 Sa se mareasca salariu cu 8 % pentru compartimen='biblio',12 % compartiment='auxiliar'
UPDATE PERSONAL_BIBLIOTECA
SET SALARIU=SALARIU*1.10 WHERE compartimen='INFORMATICA';
SET SALARIU=SALARIU*1.08 WHERE compartiment='BIBLIO';
.....
EXEMPLU 90 Folosirea cluazei CASE
UPDATE PERSOAL_BIBLIOTECA
SET SALARIU=SALARIU*
CASE compartiment
WHEN 'INFO' THEN 1.10
WHEN 'BIBLIO' THEN 1.08
ELSE 1
END
EXEMPLU 91 Se poate folosi o interogare corelata

```
UPDATE personal_biblioteca P1
SET SALARIU=SALARIU*
(SELECT CASE COMPARTIMENT
WHEN 'INFO' THEN 1.10
WHEN 'BIBLIO' THEN 1.08
ELSE 1 END
FROM personal_biblioteca p2  where p1.marca=p2..arca);
```

3.6 Comanda MERGE

Instructiunea MERGE , aparuta in ORACLE9i, permite inserarea sau actualizarea conditionata a datelor dintr-un tabel al BD

Sintaxa generala este :

```
MERGE [hint]INTO[schema.tabel_name]as alias]
USING [NUME_SCHEMA]{TABEL|VIZUALIZARE|SUBCERE}ALIAS]
ON conditie
When matched then
    Update set
  Col1={expr_1|default}.....
Col_n={expr_n}|default}.....
When  not matched then
   Insert (col_1,.....,col_n)
Values(expr1,.....,expr_n);
```

Exemplu 92 Sa se adauge articole in tabela copie_titlu astfel incat continutul sau sa includa inregistrarile titlu.

Script de merge_titlu.sql

```
**********
MERGE INTO copie_titlu c
Using titlu t
On (c.isbn=t.isbn)
When matched then
Update set
c.titlu=t.titlu,c.editura=t.editura,c.anaparitie=t.anaparitie
when not matched then
insert values(t.isbn,t.editura,t.titlu,t.anaparitie);
```

3.7 Comanda EXPLAIN PLAN

Consultarea planului de executie creat de sistem este foarte utila in optimizarea instr.SQL.

Sintaxa generala :

```
EXPLAN PLAN [SET STATEMENT_ID='text']
FOR instructiune;
```

exemplu 93 Sa se detremine planul de executie al instructiunii de actualizare a valorii cartilor care au o intrare >100;

3.8 Tranzactii

O tranzactie reprezinta o modalitate prin care se poate impacheta intr-o singura unitate de lucru o secventa de operatii privind datele din BD

3.8.1 COMMIT si ROLLBACK

EXEMPLU 94

```
SQL>ROLLBACK
Rollback complete.
```

CAPITOLUL 4 Interogari SQL

4.1 Sintaxa generala a frezei SELECT

SELECT C1,C2,.....,CN
FROM R1,R2,....RM
WHERE P
Unde ci- reprezinta coloanele (atribute sau expresii);
rj-sunt relatiile ce trebuie parcurse pt.obtinerea rezultatului
p-este predicatul simplu(conditia) sau compus ce trebuie indeplinit de tupluri(linii) pentru a fi incluse in rezultate
Exemplu 95 Selectia si proiectia
Care sunt angajatii din compartimentul 'Biblio', ordanati alfabetic ?

```
SQL>SELECT *
FROM personal_biblioteca
WHERE compartiment='Biblio'
Order by nume;
/
```

EXEMPLU 96 Sa se afiseze pe ecran informatiile din tabela edituri

```
SQL>SELECT * FROM EDITURI;
```

In Oracle, atunci cand valoarea ce se doreste a fi afisata este o constanta sau, in orice caz nu este extrasa, se recurge la o tabela dual, cu o singura linie si coloana

```
SQL>SELECT * FROM DUAL;
DUMY
.....
X
.....
```

EXEMPLU 97 Obtinerea de informatii despre data curenta

```
Sql>select sysdate from dual;
sysdate
.....
31-oct-2006
```

Predicatul de selectie (clauza where) poate contine opearatori de comparatie (>,>=,<,<=,=,#),dar si operatori BETWEEN,IS,LIKE,IS NULL ,EXISTS
EXEMPLU 98 Care sunt cartile cumparate cu pretul cuprins intre 100 si 250 ?

```
Sql>SELECT *
FROM  titluri
Where pret >=100 and pret<=250;
```

sau

```
select * from titluri where pret between 100 and 250;
```

Exemplu 99 Care sunt titlurile cartilor cu nume de autor intre Amariei ion si Marin George ?

```
Sql>select *
From titluri_autori
```

Where autor between ‘Amariei ion ‘AND ‘Marin George’;

Proiectia
Exemplu 100 Care sunt editurile din tara ?
Select disctint editura
From edituri;
/

Exemplu 101 care este nr.de telefon al editurii All.
Select telefon
From titluri
Where editura=’All’;
/
Exemplu 102 sa se listeze toate editurile din all si polirom ?
Select *
From edituri
Where editura in(‘All’,’Polirom’)
Order by editura;
/

Exemplu 103 sa se listeze numele autorului ce contine litera A pe prima poztie ?
Select *
From titluri_autori
Where upper(autor) like ‘A%’;
/
EXEMPLU 104 Sa se liteze lista autorilor a caror nume de autor apare , macar o data, litera A ?
SELECT * FROM TITLURI_AUTORI
WHERE upper(autor) like ‘%A’;

4.2 Expresii si functii sistem
4.2.1 Functii pentru sisruri de caractere
Una dintre cele mai frecvente operatiuni cu sisruri de caractere este concatenarea, care se utilizeaza fie prin operatorul ||, fie prin functia concat.
Exemplu 105
Select titlu ||’titlul cartii’|| autor as text
From titluri;

sau
SELECT CONCAT(titluri,concat
(‘titlurile cartilor’,autor)) as text
From titluri;

Exemplu 106
Sql>select editura ||’editurile din tara’||locsediu||
‘aparuta in anul’||anaparitie||’tara’
As text

From edituri;
/

Functii de conversie UPPER (toate literele vor fi convertite in majscule)
LOWER –toate literele vor fi convertite in miniscule,
INITCAP-prima litera din fiecare cuvant este majscula, iar restul litere mici
Exemplu 107
Select upper(editura),lower(editura),initcap(editura)
From edituri;

Completarea valorilor unui atribut/expresie cu un caracter pana la atingerea unei lungimi se face prin functiile LPAD(completare la stanga) si RPAD(completare la dreapta)
Exemplu 108 Interogarea urmatoarea completeaza cu spatii la stanga si la dreapta valorile atributului editura.

Select '*'||editura||'*' as editura,
'*'||lpda(editura,15)||'*' as "l_pad_15",
'*'||rpad(editura,15)||'*' as "r_pad_15:
from edituri;

operatiunea inversa, de eliminare (tundere) a spatiilor de la stanga sau de la dreapta valorii unui atribut/expresie se realizeaza prin functiile LTRIM, RTRIM
pentru eliminarea simultana a spatiilor de la stanga si de la dreapta se foloseste functira RTIM cu optiunea BOTH
Inlocuirea unui sir de caractere cu un altul intr-o expresie se utilizeaza functia REPLACE, iar pentru a extrage o portiune dintr-un sir –SUBSTR
EXEMPLU 109
SELECT autor,
replace (autor,'a','b'),
substr(autor,5,4)
from autori;
/
Functiile de numarare cuvinte si depistarea pe a cata pozitie se afla un caracter sau un sir specificat.

Exemplu 110
SELECT titlu,length(titlu),
instr(titlu,'i'),instr(titlu,'a')
from titluri;
/

4.2.2 Functii pentru valori numerice

-Cel(p)- intoarce cel mai mic intreg mai mare sau egal cu argumentul p;
-floor(p)-cel mai mare intreg mai mic sau egal cu p;
-round(p,n)-rotunjeste rezultatul expresii

-trunc(p,n)
Exemplu 111
SELECT marca,salariu,salariu/168,
ceil(salariu/168) as ceil,floor(salariu/168) as floor,round(salariu/168,2) as "round+",round(salariu/168,-2) as "round-",
trunc(salariu/168,2) as "trunc"
from pers;
/

4.2.3 Date calendaristice
In oracle9i a apaut functiile current_timestamp ce furnizeaza data so ora exacta a calculatorului , la fractiuni de secunda ce pot fi specificate.
Select current_date as "data curenta";
current_timestemp as"data si ora excacta"
from dual;
/

Pentru operatii cu date calendaristice sunt :

- Add_months(data)-aduna un numar de luni la data-argument;
- Last_day(data)-furnizeaza utlima zi din luna in care se afla data;
- Next_day(data,zi)-intoarce data primei zile(luni,marti,.....)ce urmeaza datei.

Exemplu 112
Select sysdate as "astzai",
add_months(sysdate,2) as "peste doua luni",
last_day(sysdate) "ultima_zi_a _lunii_curente",
next_day(sysdate,'tuesday') as "urmatoarea marti"
from dual;
/

Exemplu 113 Folosind functiile round si trunc
Select sysdate as "astazi",
round(sysdate,'year') as rot_an,
trunc(sysdate,'yyyy') as trunc_an,
round(sysdate,'mon') as rot_luna,
trunc(sysdate,'month') as trunc_luna,
round(sysdate,'ddd') as rot_zi,
trunc(sysdate,'ddd') as trunc_zi
from dual;
/

Exemplu 114 pentru extragerea anului/lunii/zilei dintr-o data calendaristica se utilizeaza functia EXTRACT
SELECT EXTRACT(YEAR FROM SYSDATE) AS Anul,
extract(month from sysdate) as luna,
extract(day from systimestamp) as ora,
extract(minute from systimestamp) as minute,
extract(second from systimestamp) as secunda
from dual;

/

Exemplu sa afiseze data peste 33 de zile
Select sysdate azi,
sysdate+33 as “peste 33 de zile”
from dual;
/

Exemplu 115 sa se incrementeze o data cu un interval de ordinul lunilor/anilor
Select sysdate as azi,
add_months(sysdate,26)+12 as “peste 2ani,2 luni,21 zile”,
sysdate+interval ‘2’ year+interval ‘2’ month+interval ‘12’ day as idem,
from dual;
/

In mod similar pot fi aplicate pentru atribute/variabile/constante/expresii de tipm
Exemplu 116 Pentru a afla ora excata peste 91 minute si 120 secunde de la momentul curent (cel al executiei) se poate folosi fraza SELECT
SELECT SYSTIMESTAMP AS ORA_EXACTA,
SYSTIMESTAMP+INTERVAL ‘91’ MINUTE+INTERVAL ‘120’ SECOUND AS PESTE_91 MINUTE_120_SECUNDE
FROM DUAL;
/

4.2.4 Conversii dintr-un tip in altul
Operatiunea inversa de conversie a unei date calendaristice in sir de caractere, se realizeaza prin functia to_char, respectiv conversia unui numar in sir
Exemplu 117
Select nume,
to_char(‘01/01/2006’,’dd/mm/yyyy’) as “1_decembrie”,
to+char(datasv,’dd-mm-yyyy’) as “data-sir”,
to_char(salariu,99999’) as “numar-sir”,
to_number(to_char(datasv,’mm’))
as “luna(data-sir-numar”
from personal_biblioteca
/

In oracle 9i exista o functie de tipul numtoymininterval, care converteste un numar intr-un interval year month
Exemplu 118
Select ‘5 ani ‘ as descriere,
numtoymininterval)5,’year’) as “interval’
from dual union
select ‘9 luni’,

```
numtoymininterval(7,'month')
from dual union
select '5 ani si 9 luni',
numtoymininterval(5,'year')+numtoymininterval(9,'month')
from dual union
select '5 ani si 9 luni',
numtoymininterval(5,'year')-
numtoymin(9,'month')
from dual
/
```

Exemplu 119 La fel se utilizeaza functia numtodsinterval ,dar o constanta numerica poate fi convertita dupa caz in day-zile, hour ore, mintuh –minute si secunde(secound)

4.2.5 Structuri alternative DECODE si CASE
EXEMPLU 120

```
SELECT marca,nume,colaborator,
decode(colaborator,
'n','angajat cu norma intreaga',
'd','colaborator',
'nu este este specificat ' as tipul_angajatului
From PERSONAL
```

Exemplu 121 Folosind structura CASE :

```
SELECT marca,nume,colaborator,
case colaborator
when 'n' then 'angajat_cu_norma_intreaga'
when 'd' then 'colaborator'
else 'nu este specificat'
end as tipul_angajatului
from PERSONAL
/
```

4.2.6 Jounctiuni interne
Exemplu 122 Care sunt colegii din compartimentul Biblioteca

```
Select  nume,
from PERSONAL p1, personal p2
where p1.compartiment=p2.compartiment and p2.nume='biblioteca'
```

sau

```
select nume,
from personal p1 inner join personal p2
on p1.compartimen=p2.compartiment and
p2.nume='biblio'
```

Reunine, intersectie, diferenta
Exemplu 123
Care sunt edituriile din Iasi si Bucuresti
Select * from edituri where editura=’Polirom’
Union
Select * from edituri where editura=’All’

4.2.7 Functii agregat count,sum,avg,min,max

Exemplu 124 Cati angajati numara firma ?

Functia COUNT contorizeaza valorile nenule ale unei coloane sau numarul de linii dintr-un rezultat al unei interogari

Select count(*) as nr_angajati
From personal_biblioteca;
Exemplu 125 Cate linii are produsul cartezian al tabelelor edituri si titluri ?
Select count(*)
From edituri,titluri
/
Exemplu 126 Cate edituri are tabela edituri ?
Select count(editura)
From edituri;
exemplu 127 care valoarea totala a facturii intrate in biblioteca ?
SELECT SUM(nrexem*pret) as val_totala
From titluri;
exemplu 128 care salariul mediu pe economie al angajatului din compartimentul Biblioteca ?
select count(*) as nr,sum(salariu) as suma,
avg(salariu) as sal_mediu
from personal_biblioteca
where compartiment=’biblio’
/
Exemplu 129 Care este cel mare si cel mai mic pret de intrare in biblioteca ?
Select max(pret) as cel_mare,min(pret) as cel_mic
From titluri
4.2.8 Gruparea tuplurilor :GROUP BY si HAVING
Clauza GROUP BY formeaza grupuri de linii pe baza valorilor comune ale unui atribut, iar HAVING permite selectarea anumitor grupuri de tupluri ce indeplinesc un anumit criteriu
Exemplu 130 Cate edituri lucreaza in tabela Edituri ?

Select editura,count(*) as nr_edituri
From edituri
Group by editura;

CAPITOLUL 5
Limbajul de programare PL/SQL

5.1 avantajele PL/SQL

A)POSIBILITATI PROCEDURALE-CONSTAU IN FAPTUL CA SISTEMUL SUPORTA DECLARATIILE DE VARIABILE SI CONSTANTE, FUNCTII, INSTRUCTIUNI IMPERATIVE PENTRU CELE TREI STRUCTURI FUNDAMENTALE DE PROGRAM
B)perfomante imbunatatite
c)productivitate sporita se poate construi un bloc principal PL/SQL
care apeleaza un instrument software
in practica se utilizeaza

- Sql*forms pentru dezvoltarea de aplicatii complexe
- Sql*plus
- Sql*dba pentru scierea de proceduri de explotare autmata a bazei
- Sql*reporterwriter
- Sql*menu
- Oci(oracle call interface)pentru lucrul cu rutine din biblioteci
- D)portabilitate
- E)integrarea cu SGBDR
- F)tipuri noi de date

5.2 complemntaritatea PL cu SQL
Cele doua aspecte procedural si neprocedural sunt complemtare
Aceasta complementaritate se regasesc in comenzi:

1. comenzi SQL integrate in PL/SQL: sunt cele din LMD si anumite comenzi pentru gestiunea tranzactiilor :INSERT,UPDATE,DELETE,SELECT,COMMIT,ROLLBACK,SAVEPOINT,LOCK TABLE,SET TRANSACTION,READ,ONLY
2. instructiuni proprii PL: BEGIN, END, DECLARE,l=,DECLARE..CURSOR, OPEN FETCH, CLOSE……..

5.3 Principalele functionalitati ale limbajului PL/SQL
Limbajul PL/SQL are o serie de caracteristici care definesc facilitatile sale:

- integrarea de comenzi SQL de baza
- definirea si gestiunea de blocuri de instructiuni
- gestiunea variabilelor
- gestiunea cursoarelor
- gest.execeptiilor
- structuri de control
- diferite arhitecturi

5.4 Elemente de limbaj :Elemente din SQL acceptate de PL/SQL
LIMBAJUL pl/sql ADMITE PENTRU ACCESAREA BAZEI E DATE Oracle urmatoarele elemente din SQL :

- instructiuni pentru manipularea datelor ;insert,select,update si delete
- instructiuni pentru prelucrarea tanzactiei:COMMIT, ROLLBACK SAVEPOINT

- functii
- predicate (conditiile)din clauza WHERE,toti operatori BETWEEN,IN IS,NULL,LIKE
- conditii simple sau compuse cu operatori logici AND ,OR, NOT

5.4.1 Variabilele
Variabilele din PL/SQL sunt zone de memorie utilizate pentru pastrarea unor rezultate sau pentru calcularea unor valori, care trebuie declarate inainte de utilizare in cadrul unei instructiuni.
Tipurile de variabile ce pot fi declarate de catre Oracle :
Number, char, date, boolean
Exemplul 1
Sa se declarare variabila prêt cu 8 cifre si 2 zecimale
Pret number(8,2);
atribuirea unei valori la o variabila se poate face in doua moduri:
1)prin operatoul de atribuire := , cand variabilei i se poate atribui o alta variabila, constanta, expresie sau functie
Exemplu 2 : x:=y*0.2;
2)prin clauza INTO din instructiunile SELECT sau FETCH, cand variabilei i se poate atribui o vakloare din BD

5.4.2 Constantele se declara in PL/SQL similar cu variabilele, dar se utilizeaza cuvantul CONSTANT
Exemplu 3 pi CONSTANT NUMBER(4,2):=3.14;
5.4.3 Atributele sunt asociate obiectelor PL/SQL si obiectelor bazei de date(coloane....)
Se folosesc atributele:
1.%type-care utilizeaza pentru declararea de variabilwe sau constante ce trebuie sa fie de acelasi tip cu o coloana dintr-o tabela
Exemplu 4.Variabila strada o declaram cu acelasi tip de data ca si coloana adresa din tabela pers
Strada.pers.adresa%type
2.%rowtype-care se utilizeaza pentru declararea de variabile “record’ ce trebuie sa aiba acceasi structura cu a unui anumit rand dintr-o viziune

Intreg1 prod %rowtype;

variabila intreg1 are aceeasi sttructura ca prod
3.%found
Exemplu 5 Fie declarat cursorul c1 si o instructiune fetch executata atunci se poate folosi :

if c1%found then

daca avem o instructiun insert,update,delete atunci se poate folosi if sql%found then

- %notfound-la fel ca %found
- %isopen-testeaza daca un cursor explicit a fost deschis anterior printr-o instructiune open
- %rowcount –intoarce numarul randului dupa o instructiune fetch

5.4.4 Operatori

- Operatori din PL/SQL care dau tipul expresiilor din program sunt:
- Aritmetici :+,-,*,/,**
- Logici :and, or, not
- De comparatie:= !=,<>,<=,>=
- Alti operatori is null, like (compara un sir de caractere cu un pattern, between, in,|| (concatenare)
- Expresiile rezultat pot fi aritmetice sau booleene

5.4.5 Tratarea erorilor

Tratarea erorilor este realizata in PL/SQL printr-un mechanism propriu, in bloc se declara o variabila de tip EXECPTION
Exceptiile (conditiile de eroare) pot fi definite de sistem
Exceptiile interne sunt construite autmat de cate ori un bloc PL/SQL violeaza regulile sistemului
Sintaxa generala este :
EXCEPTION
WHEN.....THEN
Instructiuni
WHEN……THEN
Instructiuni
……
END;

Urmatoarele exceptii sunt predefinite

Cursor_already_open –cursor deja deschis
…….

Exceptii definite de utilizator
Definirea exceptiilor se face in partea declarative a blocului

DECLARE
Exceptie EXCEPTION;
BEGIN
……
Construirea unei exceptii se face prin instructiunea RAISE
BEGIN
IF variabila<0 then
Raise EXCEPTIE ;
END IF;
……

EXCEPTION
WHEN exceptie
……
Intr-o rutina de prelucrare se poate folosi functiile SQLCODE care intoarce nr.erorii si SQLREM care intoarce mesajul asociat codului de eroare

5.4.6 Blocul PL/SQL
Blocul este secventa de instructiuni neexcutabile, executabile si exceptie

DECLARE
Instructiuni de declarare (neexcutabile)
Begin
Instr. Executabile (instr,proprii din PL/SQL,instr. SQL)
EXCEPTION
Tratarea erorilor
End ;

5.4.7 Gestiunea cursorului
Cursorul este o zona de memorie in care se aduce un numar de randuri egal sau diferite dintr-o tabela prin intermediul unei cereri de regasire
Exemplu 10
Declare
Cursor c1 is
Select……
Begin
……..
Fetch c1
Exit when c1%notfound;
……
End;
5.5 Concepte de baza
Blocul PL/SQL este format din trei parti

- Partea declarativa (folosind instr.DECLARE si contine instr.neexcutabile,executabila (BEGIN0
- Execptia (EXCEPTTION si contine instr.de tratarea erorilor)
- End

Partea excutabila contine la randul ei unul sau mai multe sub-blocuri (max.200 de niveluri)
Toate obiectele din PL/SQL se numesc identificatori:constante,variabile,inregistrari, cursor sau exceptie
Intr-un bloc se pot referi identificatori locali(cei declarati in blocul curent) si globali (cei declarati intr-un nivel superior)
a.)partea tranzactiilor se face folosind instr.COMMIT (finalizeaza tranzactia curenta si permanentizeaza orice schimbare)
ROLLBACK –finalizeaza tranzactia curenta si desface orice schimbare din timpul tranzactiei
SAVEPOINT-marcheaza si numeste punctul curent in prelucrarea tranzactiei

b)folosirea cursorului
pentru prelucrarea instr.pl/sql deschide o arie de context in care se pastreaza info.curenta
cursorul se defineste in partea declarativa prin instr.CURSOR,manipularea cursorului in partea executiva a blocului se face prin instr.OPEN, FETCH, CLOSE
fiecare cursor explicit are patru atribute :%notfound, %found,%rowcount, %isopen, la utilizare se scrie nume_cursor%atribut
cursorul implicit este deschis automat de Oracle in aria de context pentru a prelucra fiecare instructiune SQL.
c)crearea si accesarea inregistrarilor
o inregistrare(record)in PL/SQL este o variabila care contine un grup de variabile elementare, numite campuri

exemplu crearea unei inregistrari in PL/SQL se poate face folosind %rowtype, daca se declara un cursor explicit care regaseste coloane

```
declare
cursor c1 is select codp,denp from prod;
intreg c1%rowtype;
```

5.6 Instructiunile
Atribuirea in PL/SQL se realizeaza cu ajutorul operatorului de asignare:=

Var:=expresie;

exemplu
nume:=’ionescu’;

begin este o instructiune care se incepe partea executabila a unui bloc
close –inchide un cursor

close nume_cursor;
commit-permanentizeaza orice schimbare in Bdexecuta de catre tranzactia curenta
COMMIT[WORK];
DECLARE-marcheaza inceputul unui bloc
DECLARE
Instr. Neexecutabile de declarare

Pentru a declara variabile si constante se folosesc tipurile :
NUMBER[(precizi[,scala];
exemplu
x1 number(4,2);
x2 constant number:=10;
x3 number;
pentru date si sir de caractere
CHAR[(lungime)] ;
Exe>nume CHAR(35) ;

 ISSN 2069-7988
ISSN-L 2069-7988

VARCHAR2 LA FEL
DATE pentru date
Boolean pentru variabile boolene (true,false,null)
%type este un atribut pentru declarea de variabile si constante
Ex. cod1 cod%type ;

%rowtype –pentru declarea var. de variabile de tip inregistrare

2.Pentru a declara un cursor se utilizeaza urmatoarea sintaxa :

CURSOR nume-cursor[nume-param tip _param]
IS SELECT,.....

Unde nume-cursor sete numele cursorului;
nume-parm=este un identificator care va fi folosit in instr.sql select
tip_parm=poate fi char,number,date,Boolean

exemplu 15
se declara cursorul cu numele c1 construit prin selectarea coloanelor
edityura,locsediu din tabela edituri

cursor c1 is select editura,locsediu from edituri
where editura=’All’;

pentru a declara exceptia se foloseste :
nume_exceptie EXCEPTION;

DELETE STERGE TOATE RANDURILE SPECIFICATE DINTR-O TABELA SAU VIZIUNE :

delete[from TABELA]WHERE CONDITIE]|CUURENT OF CURSOR

end –incheie o serie de instr.
END[NUME-ETCHETA];
END LOOP;
END IF;

EXCEPTION marcheaza ineputul unei partii execptiei (pentru tratarea erorilor
Sau explicit prin instr.RAISE-in partea executive

EXCEPTION
WHEN nume-execptie |other then INSTR.;
……

Unde nume_exceptie este un nume definit de utilizator sau exceptii predefinite de PL/SQL:
CURSOR_ALREADY,,,,,,,,,,,,,,,,,,,,,

EXCEPTION_INIT atribuie un nume de cod de eroare Oracle
PRAGMA EXCEPTION_INIT (nume_exceptie, cod_eroare)

Unde PRAGMA este un cuvant rezervat care desmneaza o directiva de compilare
Exit- forteaza iesirea din structura nesceventiala curenta

EXIT [nume-etcheta][when conditie] ;

Exemplu 16
Begin
….
If x>10 then
Instructiuni
Else exit ;
End if;
……
For I in 1….10 loop
……
Exit unu when……
End loop unu

Fetch acceseaza (prelucreaza) urmatorul rand de date din multimea active
FTECH nume_cursor INTO lista_var|nmumar intreg;

Exemplu 17

Declare
Cursor c1 is select editura,locssediu from edituri;
Intreg c1%rowtype
--variabiiala intreg la fel ca si cursorul c1
.....
Begin
Open c1
--deschid cursorul
.....
Loop
Fetch c1 into intreg;
--accesz cate un rand de date
--in ciclu pana la terminarea acestora
Exit when c1%notfound;
.....
End loop;
Close c1;
End

GOTO salt neconditionat la o instr.executabila

Goto nume_etcheta;
exemplu 18
begin
.....
Goto actualiz;
--salt la blocul de actualizare
Begin
Update...
.....
End actualiz;
.....
End;

IF –o structura alternative
Forma generala
If conditie then
Instr.....
[elseif conditie1 then instr.....
  [elseif conditie2 then instr……]
[else instr….]
End if ;

Insert –adauga inregistrari in tabela

INSERT INTO nume-tabela[col1,……..]
Values(expr1,.....)|SELECT.....;

exemplu 19
insert into edituri select editura,locsediu from edituri1
where editura='All';
……
Lock table – blocheaza una sau mai multe tabele
Forma generala
Lock table nume-tabela1,..... in mod mode[nowait];
Unde nume-tabela –reprezinta numele tabelelor care vor fib locate
Mod=modul de blocare :row,share,row exclusiv,share,update
Nowait – specifica ca se returneaza controlul unui alt utilizator care a blocat tabela

Exemplu lock table edituri in share mode;

Loop –structura repetitive
Forma generala
[<<etchieta>>]while conditie\[for par_numeric][ar_cursor]loop
Instr……
End loop [<<etcheta>>];

 ISSN 2069-7988
ISSN-L 2069-7988

Var in[reverse] vi,vf
variabila de indexare var nu trebuie declarata anterior va lua valorile de la vi pana la valorea finala vf
nume_intreg in nume_cursor[(par1,.....)]|SELECT .....

null;

open nume-cursor[(par1,.....)];

exemplu
begin
open c1;
open c2(mcomp,'biblio');
…….
End;

Raise 0-opreste executia unui bloc
Raise [nume-exceptie];

Rollback desface(anuleaza)explicit unele sau toate schimbarile din baza de date
Rollback[work|to[savepoint] nume-salvare;

Savepoint-marcheaza si numeste punctele curente in prelucrarea tranzactiei

Select lista-sel into nume-var|nume-intreg
From nume-=tabela rest_sel;

Exemplu 20 Primul bloc PL/SQL
--ACEST BLOC NU FACE APROAPE NIMIC
DECLARE
/*PRIMA SECTIUNE ESTE CEA A DECLARATIILOR ,VARIABILE, CURSOARE,EXCEPTII */
Prima_variabila integer;
A_doua_var VARCHAR2(35);
A_TREIA_VAR DATE;
Ulitma_var BOOLEAN;

BEGIN
--in aceasta sectiune se scriu comenzile effective
DBMS_OUTPUT.PUT_LINE('Va salut !');
--in SQL este neceasara comanda SET SERVEROUTPUT ON

END;
/

Acest script lanseaza direct din SQL*PLUS

Pentru lansarea in executie a unui script se utiliezaza comenzile START sau @
Sql>@c:\biblio\migrare_oracle\primul.sql
/
Sau crearea unui dosar in E :\aplicatii_oracle
Dupa lanasare apare pe ecran mesajul Va salut !

5.7 Tipuri de date PL/SQL.Domeniul de vizilibilitate al variabilelor
Schema cu tipurile de date

Exemplu 21 Bloc inclus.Domeniul de vizibilitate
Exemplu21_bloc_inclus.sql

```
--blocul principal
DECLARE
A integer :=12 ;
B varchar2(20) ;
C date;

Begin
B:='Informatica este stiinta'
C:=to_date('31/10/2006','dd/mm/yyyy');

Dbms_output.put_line('');
Dbms_output.put_line('La inceputul blocului principal');
Dbms_output.put_line('a='||a);
Dbms_output.put_line('b=||b);
Dbms_output.put_line('c='||c);

--blocul secundar

Declare

B number(12,2);
C varchar2(40);
D date;

Begin
B:=123;
D:=to_date('30/11/2006','dd/mm/yyyy');
Dbms_output.put_line('');
Dbms_output.put_line('la inceputul blocului secundar');
Dbms_output.put_line('a='||a);
Dbms_output.put_line('b='||b);
Dbms_output.put_line('c='||nvl(c,'c este null'));
Dbms_output.put_line('d='||d);
End;

--revenire la blocul principal
```

```
Dbms_output.put_line(‘’);
Dbms_output.put_line(‘la revenirea blocului principal’);
Dbms_output.put_line(‘a=’||a);
Dbms_output.put_line(‘b=’||b);
Dbms_output.put_line(‘c=’||c);

End;
```

5.8 Structuri alternative si repetitive. Exemple

Exemplu 22_ecuatie_grd.ii.sql

```
--blocul pentru rezolvarea ecuatiei de grd.ii ax**2+b*x+c=0
DECLARE
A integer :=0 ;
B integer :=10 ;
C integer:=12;
Delta number(16,2);
X1 number(16,6);
X2 number(16,6);
BEGIN
--ec.de grd.ii ?
If a=0 then
   If b=0 then
      If c=0 then
            Dbms_output.put.line(‘nedeterminare!’);
         Else
            Dbms_output.put.line(‘imposibil !!!’);
     End if;
Else
      Dbms_output.put_line(‘ecuatia de grd.i !’);
X1:=c/b;
Dbms_output.put_line(‘x=’||x1);
End if;
Else
Delta:=b**2-4*a*c;
If delata >0 then
   X1:=(-b-sqrt(delta))/(2*a);
  X2:=(-b+sqrt(delta))/(2*a);
  Dbms_output.put_line(‘x1=’||x1||’,x2=’||x2);
Else
If delta:=0 then
   X1:=-b/(2*a);
Dbms_output.put_line(‘x1=x2=’||x1);
Else
Dbms_output.put_line(‘radacinle sunt complexe !!!’);
End if;
End if;
End if;
```

```
End;

Executia programului este :
SQL>@e:\aplicatii_sql\exemplu_22.sql;
Exemplu 23 Sa se foloseasca structura CASE
EXEMPLU 24 Blocul PL/SQL pentru popularea tabele pontaje pentru o luna
--popularea cu inregistrari pentru o luna dintr-an a tabele pontaje
DECLARE
An salarii.an%type :=2006 ;
Luna salarii.luna%type:=1;
Prima_zi date;--variabila care stocheaza data de 1 a lunii
Zi date;
BEGIN
Prima_zi:=TO_DATE('01/'||luna||'/'||an,'DD/MM/YYYY');
Zi:=prima_zi;

While zi<=last_day(prima_zi) loop
   If rtrim(to_char(zi,'DAY')) IN ('Saturday','sunday') then
    --e zi nelucratoare (sambata /duminica)
      NULL;
   ELSE
     INSERT INTO pontaje(marca,data)
Select marca,zi from personal;
End if;
--se trece la ziua urmatoare
End loop;
Commit;
End;


Exemplu 25 Un alt mod de redactare a structurii repetitive

--popularea cu inregistrari pentru o luna dintr-an a tabele pontaje
DECLARE
An salarii.an%type :=2006 ;
Luna salarii.luna%type:=1;
Prima_zi date;--variabila care stocheaza data de 1 a lunii
Zi date;
BEGIN
Prima_zi:=TO_DATE('01/'||luna||'/'||an,'DD/MM/YYYY');
Zi:=prima_zi;

Loop
  Exit when zi<=lat_day(prima_zi) ;

 If rtrim(to_char(zi,'DAY')) IN ('Saturday','sunday') then
   --e zi nelucratoare (sambata /duminica)
     NULL;
```

```
   ELSE
      INSERT INTO pontaje(marca,data)
Select marca,zi from personal;
End if;
--se trece la ziua urmatoare
End loop;
Commit;
End;
```

Exemplu 26 ce de-a treia varianta prezentata foloseste instructiunea FOR….NEXT (ENDFOR DIN VFP)

```
--popularea cu inregistrari pentru o luna dintr-an a tabele pontaje
DECLARE
An salarii.an%type :=2006 ;
Luna salarii.luna%type:=1;
Prima_zi date;--variabila care stocheaza data de 1 a lunii
Ultima_zi DATE;
Numar_ultima_zi PLS_INTEGER;

Zi date;
BEGIN
Prima_zi:=TO_DATE('01/''||luna||''/'||an,'DD/MM/YYYY');
Ulitma_zi :=last_day(prima_zi);
Numar_ultima_zi:=to_number(to_char(ultima_zi,'dd'))
/*bucla se repeat pentr i=1-31 (30,28,29) */
FOR I IN 1..numar_ultima_zi LOOP
Zi:=prima_zi;+i-1;
If to_char(zi,'DAY') in('SAT','SUN') THEN
--e zi nelucratoare
Null;
Else
Insert into pontaje(marca,data)
Select marca,zi from personal;
End if;
--se trece automat la ziua urmatoare
End loop;
Commit;
End;
/
Forma generala a instr.FOR….NEXT   este :
FOR variabila_contor IN valoare_initiala..valoare_finala   LOOP
```

5.9 Exceptii

Daca s-a lansat in executie de doua sau de trei ori un script pentru inserarea de articole intr-o tabela, atunci apare o eroare

ERROR-00001……
Are un nume predefinit si anume :dup_val_on_index
Exemplu 27 Bloc inclus si tratarea unei eceptii

```
 --popularea cu inregistrari pentru o luna dintr-an a tabele pontaje
DECLARE
An salarii.an%type :=2006 ;
Luna salarii.luna%type:=1;
Prima_zi date;--variabila care stocheaza data de 1 a lunii
Zi date;
BEGIN
Prima_zi:=TO_DATE('01/'||luna||'/'||an,'DD/MM/YYYY');
Zi:=prima_zi;

While zi<=last_day(prima_zi) loop
   If rtrim(to_char(zi,'DAY')) IN ('Saturday','sunday') then
    --e zi nelucratoare (sambata /duminica)
      NULL;
   ELSE
     INSERT INTO pontaje(marca,data)
Select marca,zi from personal;

EXCEPTION—se preia eventuala violare a cheii primare
WHEN DUP_VAL_ON_INDEX THEN
--se sterg inregistrarile pentru ziua curenta
DELE FROM PONTAJE where DATA :=ZI ;
--apoi se reinsereaza inregistrarile

INSERT INTO pontaje(marca,data)
Select marca,zi from personal;
END;--se elimina blocul inclus

End if;
--se trece la ziua urmatoare
End loop;
Commit;
End;
```

5.10 Cursoare
Executia comenzilor SQL si stocarea informatiilor procesate presupun folosirea de catre SGBD a unor zone de lucru speciale
Cursoarele oracle sunt :implicite, explicite
5.10.1 Cursoare implicite sunt create automat de sistem la excutia comenzilor DML(INSERT,UPDATE,DELETE)

In urma executiei unei comenzi SQL de actualizare,Oracle pastreaza o serie de informatii despre rezultate :
Exemplu 28 Scriptul urmatoar contine blocul PL/SQL care mareste salariul cu 30 de ori, dar numai pentru angajati cu peste un numar de ani vechime

```
DECLARE
Ani_etalon pls_intger :=50 ;
Numar pls_integer ;
Begin
/*se mareste cu 30 ori…..*/
UPDATE PERSONAL
SET salariu=salariu+30
Where months_between(sysdate,datasv)/12>=ani_etalon ;
If SQL%FOUND THEN
DMBS_OUTPUT.PUT_LINE('EXISTA CEL PUTIN UN ANGAJAT CU PESTE >45');
Numar:=sql%rowcount;
Dbms_output.put_line('numarul='||numar);
Else
Dbms_output.put_line('Nu exista nici un angajat !');
End if;
End;
/
```

5.10.2 Cursoare explicite
Schema de princpiu a unui cursor explicit
-declarea cursorului printr-o fraza SELECT(CURSOR nume IS SELECT……)
-declarea variabilelor in care va fi stocata o linie a cursorului;
-Deschiderea cursorului OPEN;
-incarcarea urmatoarei linii din cursor (FETCH);
-structura de ciclare din cursor.

```
DECLARE
--cursorul se declara printr-o fraza select
CURSOR c_cursor IS
    SELECT……..
--se declara variabila compusa ce va stoca o linie a cursorului

Rec_cursor c_cursor%ROETYPE ;
…….
Begin
……
OPEN c_cursor
Fetch c_cursor into rec_cursor

While  c_cursor%FOUND LOOP
……..
……..
--corpul buclei
```

```
Fetch c_cursor into rec_cursor   /*se incearca incarcarea urmatoare linii din cursor,
daca nu s-a putut, rezulta ca inregistrarile cursorului sunt epuizate, iar
c_cursor%FOUND are valoarea logica FALSE */
End loop
Close C_CURSOR
…….
END;
/
```

EXEMPLU 29 Cursor explicit pentru marirea salariilor orare ale unor angajati

```
Declare
Cursor c_salariati is
Select marca,nume,compartiment,salariu,
Trunk(months_between(sysdate,datasv)/12,0)
As ani_cevhime
From personal order by compartiment,5 desc;

Rec_salariati%rowtype;
V_compart personal.compartiment%type:=’xyz’;
--variabila v_compart va stoca codul compartimentului
V_ani integer :=99 ;
Numar integer :=1 ;
Begin
--se deschide cursorul
Open c_salariati;
Fetch c_salariati into rec_salariati;

--se stabilesc conditiile pentru iteratie
WHILE c_salariati%FOUND LOOP
   If c_recsalariati<>v_compart then
     Numar:=1;
V_compart:=rec_salariati.compartiment;
V_ani:=rec_salariati.ani_vechime;
Update personal
Set salariu=salariu+salariu*0.1
Where marca=rec_salariati.marca ;
Else
If numar>3 and rec_salariati.ani_vechime<v_ani then
Update personal
Set salariu=salariu+salariu*0.1
Where marca=rec_salariati.marca;
Numar:=numar+1;
V_ani:=rec_salariati.ani_vechime;
End if;
End if;

Fetch c_salariati into rec_salariati;
End loop;
```

```
Close c_salariati;
End;
/
```

A doua varianta are urmatoarele avantaje :

- Nu mai este necesara declararea explicita a variabilei in care se citeste o inregistrare din cursor;
- Cursorul nu mai trebuie deschis explicit si nici inchis;
- Incarcarea urmatoare inregistrari se face automat la reluarea buclei.
- A doua schema generala de folosire a unui cursor explicit

```
Declare
Cursor c_cursor is
Select…….
…….
Begin
…….
For rec_cursor in c_cursor loop
Corpul buclei
……..
End loop
,,,,,,,,,,
End;
/
```

Exemplu 30 a doua varianta de lucru cu un cursor explicit c_salariati

```
Declare
Cursor c_salariati is
Select marca,nume,compartiment,salariu,
Trunk(months_between(sysdate,datasv)/12,0)
As ani_cevhime
From personal order by compartiment,5 desc;

V_compart personal.compartiment%type:=’xyz’;
--variabila v_compart va stoca codul compartimentului
V_ani integer :=99 ;
Numar integer :=1 ;
Begin
For rec_salariati in c_salariati loop
If c_recsalariati<>v_compart then
    Numar:=1;
V_compart:=rec_salariati.compartiment;
V_ani:=rec_salariati.ani_vechime;
Update personal
Set salariu=salariu+salariu*0.1
Where marca=rec_salariati.marca ;
Else
```

 ISSN 2069-7988
ISSN-L 2069-7988

```
If numar>3 and rec_salariati.ani_vechime<v_ani then
Update personal
Set salariu=salariu+salariu*0.1
Where marca=rec_salariati.marca;
Numar:=numar+1;
V_ani:=rec_salariati.ani_vechime;
End if;
End if;
--incarcarea urmatoare inregistrari se face automat

End loop;

End;
/
```

 ISSN 2069-7988
ISSN-L 2069-7988

CAPTILOUL 6
Limabjul SQL comenzi de lucru in VFP

1.Principalele clause ale frazei SELECT
A1)Selectia si Proiectia
Forma generaa a comenzii SELECT
SELECT C1,C2,.....CN
FROM R1,R2,.....RM
WHERE P
Prin executia unei fraze SELECT se obtine un rezultat de forma tabelara.Aceasta poate fi lista, o tabela temporara
Unde ci=reprezinta coloanele (care sunt atribute sau expresii)
Rj=sunt relatiile ce trebuie parcurse pentru obtinerea rezultatului
P=este predicatul (conditia) simplu sau compus ce trebuie indeplinit de tupluri
Exemplu 1
Sa se listeze toate editurile din tara ?
SELECT * FROM EDITURI;
EXEMPLU 2 Sa se elimine dublurile ?
Select distinct editura,locsediu from edituri;
Exemplu 3 Care sunt este editura Polirom ?
SELECT * FROM EDITURI;
WHERE EDITURA=’POLIROM’;

EXEMPLU 4 Care sunt facturile emise in perioada 10-15 septembrie 2006 ?
Select * from titluri;
Where dataprim >=’2006/09/10’ and dataprim<=’2006/09/15’;

Proiectie
Exemplu 5 Care sunt denumire-clasa,clasa din tabela clasificare zecimala universala ?
Select dendiviz,clasa;
from czu;

exemplu 6 Care sunt cartile primate de la editura All ?
SELECT titlu,autor,isbn,adresa;
From titluri;
Where editura=’All’;

A2)Reuniune,intersectie,diferenta,produs cartezian
Forma generala pentru reuniune
Select *
From r1
Union
Select *
From r2
Exemplu 7 Care sunt denumirele editurilor si adresa ale editurilor polrom si All ?
SELECT EDITURA,ADRESA,LOCSEDIU
FROM EDITURI

```
WHERE EDITURA='All'
Union
Select editura,locsediu,adresa
From edituri
Where editura='Polirom';
```

intersectia
forma generala este :

```
select *
from r1
intersect all
select *
from r2
```

exemplu 8 Care sunt facturile cartilor care apar simultan si in factura 1000 si 2000 ?

```
SELECT TITLU
FROM CARTI
WHERE NRFACT=1000
INTERSECT
SELECT TITLU
FROM CARTI
WHERE NRFACT=2000;
```

EXEMPLU 9 cu diferenta a doua tabele
Forma generala este :

```
select *
from r1
except
select *
from r2
```

sau oracle ofera solutia cu MINUS

exemplu10 Care sunt facturile cartilor care apar in factura 1000 , dar nu apar in factura 2000 ?

```
SELECT TITLU
FROM CARTI
WHERE NRFACT=1000
MINUS
SELECT TITLU
FROM CARTI
WHERE NRFACT=2000;
```

Produs cartezian
Forma generala este :

```
SELECT * FROM R1,R2;
```

EXEMPLU 11 Sa se listeze informatiile editura,locsediu,titlu,autor din tabele edituri,titluri ?
SELECT EDITURA,LOCSEDIU,TITLU,AUTOR ;
FROM EDITURI,TITLURI;
EXEMPLU 11 Sa se listeze cartile imprumutate respectiv cartile nerestituite ?
Select nrpermis,dataimp aas data_imprumut;
titlu,autor,datarest as data_restituire;
from imprumut;

exemplu 12
select nrpermis,numepren,titlu,autor,dataimp+14 as data_restituire;
from imprumut;

exemplu 13 cu functii de tip data calendaristica
select nrpermis,autor,dataimp as data_imprumut,;
gomonth(dataimp,2) as scxandenta;
from imprumut;

exemplu 14
select nrpermis,titlu,dataimp;
gomonth(dataimp,14)+15 as o_data_viitoare ;
from imprumut;

a3)optiunea ORDER BY
EXEMPLU 15 Care sunt editurile din tabela edituri in ordinea alfabetica ?
SELECT *
FROM EDITURI;
ORDER BY EDITURA;

EXEMPLU 16 Sa se obtina in ordinea descrescatoare a editurilor si in ordinea crescatoare a localitatilor ?
Select editura,locsediu,adresa;
from edituri;
order by editura desc,locsediu asc;

a4)Operatorii between,like,in
operatorul BETWEEN
Este util pentru definirea intervalelor de valori
Exemplu 17 Care sunt facturile intrate in biblioteca in perioada 10-15 sept.2006 ?
Solutia data in VFP

SELECT *;
FROM CARTI;
WHERE DATEPRIM BETWWEN {10/09/2006} AND {15/09/2006};

EXEMPLU 18 Sa se listeze in ordinea editurilor localitatile editurilor ?
Select editura,locsediu;
from edituri;

where editura betwwen ‘Polirom’ and ‘All’;
order by editura,locsediu;

operatorul LIKE
EXEMPLU 19 Sa se listeze informatiile din tabela Edituri incepand cu prima litera A ?
SELECT *;
FROM EDITURI;
WHERE EDITURA LIKE ‘%A’;

Operatorul LIKE permite compararea unui atribut cu un literal folosind o masca construita cu ajutorul specificatorilor %
Exemplu 20 Ce autori au litera V ?
Select * from titluri_autori ;
Where autor like ‘_V%’ ;

Operatorul IN
FORMATUL GENERAL :
EXPRESIE1 in (EXPRESIE2,EXPRESIE3);
Exemplu 21 Care sunt editurile All si Polirom ?
Select *;
From edituri;
Where editura=’All’ OR editura=’Polirom’;
Order by editura;

2.Theta- si echi-jonctiunea
Forme generala :
select *
from r1. inner join r2 on r1.a>=r2.e;
exemplu 22 Sa se listeze pentru fiecare editura :editura,locsediu,titlu,autor
select editura,titlu,autor;
from edituri,titluri;
where edituri.editura=titluri.editura;

varianta a-2-a

SELECT EDITURA,LOCSEDIU,TITLU,AUTOR;
FROM EDITURI INNER JOIN TITLURI ON;
EDITURI.EDITURA=TITLURI.EDITURA;

EXEMPLU 23 Care sunt editurile cu numele Gil din tabela titluri_autori, titluri ?
Select editura,titlu,autor,isbn;
From titluri inner join titluri_autori on;
Titluri.isbn=titluri_autori.isbn;
where editura=’Gil’;

3.Sinonome locale si jonctiunea unei tabele cu ea insasi
Exemplu 24 Ce facturi au fost intrate in acceasi zi cu factura 1000 ?
Select f1.nrfact ;
From inventar i1,inventar i2 ;
Where i1.nrfact=f1.nrfact and i1.nrfact=1000;

4.Functii –agrgat :COUNT,SUM,AVG,MIN,MAX
Formatul general al unei fraze SELECT ce contine functii-agregat este :
SELECT FUNCTIE-PREDEFINITA1,FUNCTIE-PREDEFINITA2,…….
FROM LISTA-TABELE;
WHERE CONDITIE;

Functia COUNT
Functia COUNT contorizeaza valorile nenule ale unei coloane sau numarul de linii dintr-un rezultat al unei interogari.
Exemplu 25 Cate edituri are societatea ?
Select count(*) as nredituri;
from edituri;

Exemplu 26 Cate linii are produsul cartezian al tabelelor edituri si titluri ?
SELECT COUNT(*);
FROM EDITURI,TITLURI;

EXEMPLU 27
Pentru cati clienti se cunoaste adresa ?
SELEC COUNT(ADRESA) AS NR.CLIENT;
FROM CLIENTI;

EXEMPLU 28 Cate facturi au intrat pe data 10.09.2006 ?
Select count(nrfact) as nrfacturi ;
From inventar;
Where dataprim={10/09/2006};
Functia SUM
EXEMPLU 29 Care este valoarea totala a unei facturi de intrare in biblioteca ?
SELECT SUM(NREXEM*PRET) AS VALTOTALA;
FROM TITLURI;

Functia AVG –Calculeaza media aritimetica a unei coloane intr-o tabela prin divizarea sumei valorilor respective
Exemplu 30
Care valoarea medie a cartilor intrate in biblioteca ?
SELECT AVG(NREXEM*PRET) AS VAL_MEDIE;
FROM TITLURI;

Functiile MIN si MAX
Exemplu 31 Care este editura cu prima, ultima denumire ?
SELECT MIN(EDITURA),MAX(EDITURA);
FROM EDITURI

EXEMPLU 32 Care este cel mai mic respectiv cel mare pret la care a intrat in biblioteca ?

```
Select min(pret),max(pret) ;
from titluri
```

5 Gruparea tuplurilor GROUP si HAVING
Clauza GROUP BY
FORMA GENERALA ESTE :

```
Select coloana1,……coloana n
From tabela
GROUP BY coloana-de-regrupare;
```

Exemplu 33 Care este valoarea fara TVA a fiecarei factura ?

```
SELECT NRFACT,SUM(NREXEM*PRET) AS VALFARATVA;
FROM TITLURI;
GROUP BY nrfact
```

Exemplu 34 Care este valoarea totala a intrarilor de carti pentru fiecare zi ?

```
Select dataprim,sum(nrexem*prêt*(1+proctva)) as valtotala;
From carti;
Group by dataprim
```

Exemplu 35 Care este cea mai mare valoare a unei facturi ?

```
SELECT MAX(SUM(NREXEM9PRET)) AS VALMAXTOTALA;
FROM CARTI;
GROUP BY NRFACT
```

Clauza HAVING- ESTE where-ul ce opereaza la nivel de grupuri
Forma generala este

```
Select col1,col2,.....coln
From tabela
Group by col-de-regrupare
Having caracteristica-de-grup
```

Exemplu 36 Care sunt zilele in care s-au intocmit cel putin trei facturi ?

```
SELECT DATAPRIM,COUNT(*) AS NR_FACTURI;
FROM TITLURI;
GROUP BY dataprim;
Having count(*)>=3
```

Exemplu 37 Care sunt editurile pentru care intrarile de carti in biblioteca depasesc 10milioane ?

```
Select editura as editua,
sum(nrexem*pret) as valoare_intrata;
from titluri t,titluri_autori ta;
```

```
where t.isbn=ta.isbn;
group by editura;
having sum(nrexm*prêt)>10000000
```

5.Prelucrarea valorilor nule
Exemplu 38 Pentru care dintre clienti (cititori) nu se cunoaste adresa ?
Solutia cvasigenerala se bazeaza pe utilizarea operatorului IS NULL, care extrage toate valorile NULL pentru un atribut.

```
SELECT *;
FROM CLIENTI;
WHERE adresa IS NULL
```

Valoarea null nu confunda cu valoarea 0, sau cu valoarea=' '
In versiunile vechi era problema introducerii :

```
select * from clienti where adresa=" ";
sau select * from clienti where empty(adresa);
```

Exemplu 39 Sa se creeze tabele sporuri si personal in ORACLE

```
Drop table sporuri;
drop table personal;

create table personal (
marca integer constraint ok_personal primary key,
nume varchar2(35),
datanast date,
compart varchar2(20),
salariu integer,
marcasef integer constraint fk_personal references personal(marca)
);

Create table sporuri (an integer,
Luna integer,
Marca integer refrences personal(marca),
sporvechime integer,
altesporuri integer,
primay key(an,luna,marca)
);
```

Exemplu 40 Care sunt persoanele si lunile pentru care nu s-a calculat (nu se cunoaste)sporul pentru conditii periculoase ?

```
SELECT SPORURI.MARCA,nume,compart,an,luna
From personal,asporuri
Where personal.marca=sporuri.marca and altesporuri IS NULL;
```

EXEMPLU 41 Care sunt angajati si lunile in care acestia nu au primit spor pentru conditii periculoase ?

```
SELECT sporuri.marca,nume,compart,an,luna
```

From personal,sporuri
Where personal.marca=sporuri.marca and altesporuri=0
Order by an,luna,nume

Exemplu 42 Sa se afle lista persoanelor angajate care sunt nascuti de 01.decembrie 1990 si care dupa aceasta data ?
Select *
From personal
Where datanast<'01-12-1990'
Si dupa
Select *
From personal
Where datanast>+'01-12-1990'

Exemplu 43 Care totalul sporurilor de vechime) pentru luna decembrie 1989 ?
Select sum(sporurivechime) as total_sporuri
From sporuri
Where an=1989 and luna=12;

In Oracle si VFP exista o functiwe NVL de la Nullvalue)
SELECT sporuri.marca,nume,compartiment,
nvl(sporvechime,0)+nvl(altesporuri) as totalsporuri
from personal,sporuri
where personal.marca=sporuri.marca and an=1989 and luna=12;

6.Jonctiunea externa
Standardul SQL-92 introduce operatorii jonctiunii externe :

- Left outher join –pentru jonctiune externa la stanga
- Rigth outher join-pt.jonctiune externa la dreapta
- Full outher join –jonctiune totala (in ambele directii)

Fome generale :
Jonctiune externa la stanga :
Select *
From r1 left outher join r2 on r1.c=r2.c
Jonctiune externa la dreapta
Select *
From r1 right outher join r2 on r1.c=r2.c
Jonctiune totala
Select *
From r1 full outher join r2 on r1.c=r2.
Exemplu 44 Care sunt editurile care nu are nici o editura ?
SELECT *
FROM EDITURA LEFT OUTHER JOIN TITLURI
ON EDITURI.EDITURA=TITLURI.EDITURA
WHERE TITLURI.EDITURA IS NULL

 ISSN 2069-7988
ISSN-L 2069-7988

CAPITOLUL 7. APLICATII INFORMATICE UTILIZAND LIMBAJUL SQL

7.1. Aplicatie informatica pentru activitatea de salarizare

1) Folosindu-se instructiunile SQL, sa se creeze tabelele DatePers, DateSal, Impozitar, Pontaj, SporVechime, Taxe, Deduceri.

CREATE TABLE DatePers

codang number (5) primary key,
nume varchar2 (35),
cnp varchar2 (13),
datan date,
adresa varchar2 (30),
localitate varchar2 (15),
telefon varchar2 (12)

CREATE TABLE DateSal

codang number(5) references DatePers (codang),
functia varchar2 (10),
salbaza number (15),
persintr number (2),
vechime number (3),
codsef number (5) references DatePers (codang)
);

CREATE TABLE Impozitar

linie number (5) primary key,
dela number (15),
panala number (15),
suma number (15),
procent number (3)

CREATE TABLE Pontaj

```
codang      number (5) references DatePers(codang),
luna        number (3),
zilelucr    number(3),
orezi       number(3),
zileco      number(3),
zilecm      number(3),
orelucrate  number(4),
        constraint pk primary key(codang,luna)
```

```
CREATE TABLESporVechime

        nr          number (3) primary key,
        dela        number (3),
        panala      number (3),
        procent     number (3)
```

```
CREATE TABLE Taxe
(
        den         varchar2 (10) primary key,
        procent     number (2),
        cotamax     number (15)
```

```
CREATE TABLE Deduced

den         varchar2 (30) primary key,
cotasuma    number(15),
cotaproc    number(2),
cotamax     number (15)
```

2) Sa se Tncarce cu date tabelele create.

```
SQL> DELETE FROM DatePers;
        INSERT INTO DatePers VALUES
        (100, Ion Ion', '1234567890100', '10-JAN-1970', 'Mangaliei 100',
        'Constanta', '0722123456');
        INSERT INTO DatePers VALUES
```

(200, 'Popescu Ion', '1234567890200', '10-FEB-1975', 'Tomis 232', 'Constanta', '0744123456');
INSERT INTO DatePers VALUES
(300, lonescu Gheorghe', '1234567890300', '10-MAR-1980', 'Ferdinand 48', 'Mangalia', '0788123456');

SQL> DELETE FROM DateSal;
INSERT INTO DateSal VALUES (100, 'Ec', 5000000,1,10, 300); INSERT INTO DateSal VALUES (200, 'Inginer', 6500000, 2, 5, 300); INSERT INTO DateSal VALUES (300, 'Director', 15000000, 0,15, null);

SQL> DELETE FROM Impozitar;
INSERT INTO Impozitar VALUES (1, 0, 2100000, 0,18);
INSERT INTO Impozitar VALUES (2, 2100001, 5200000, 378000,23);
INSERT INTO Impozitar VALUES (3, 5200001, 8300000,1091000,28);
INSERT INTO Impozitar VALUES (4, 8300001,11600000,1959000,34);
INSERT INTO Impozitar VALUES (5, 11600001, 99999999999, 3081000, 40);

SQL> DELETE FROM Pontaj;
INSERT INTO Pontaj VALUES (100, '1', 22, 8, 3, 0,170);
INSERT INTO Pontaj VALUES (200, 'V, 30, 8, 5,10, 200);
INSERT INTO Pontaj VALUES (300, 'V, 22, 8, 0, 0,176);

SQL> DELETE FROM SporVechime;
INSERT INTO SporVechime VALUES (1, 0, 3, 5);
INSERT INTO SporVechime VALUES (2, 4,10,10);
INSERT INTO SporVechime VALUES (3,11, 20,15);
INSERT INTO SporVechime VALUES (4, 21, 40, 20);

SQL> DELETE FROM Taxe;
INSERT INTO Taxe VALUES ('CASS', 6.5, 999999999999999);
INSERT INTO Taxe VALUES ('CAS', 9.5,15000000); INSERT INTO Taxe VALUES ('Somaj', 1, 999999999999999);

SQL> DELETE FROM Deduced;
INSERT INTO Deduced VALUES ('Deducere de baza', 1800000, 0, 1800000);

 ISSN 2069-7988
ISSN-L 2069-7988

INSERT INTO Deduced VALUES ('Deducere suplimentara', 0, 0.5, 3600000);
INSERT INTO Deduced VALUES ('Cheltprofesionale', 0,15, 270000);

Interogarea tabelelor bazei de date

1) Sa se afi§eze informatiile despre angajatii firmei.

SQL> SELECT * FROM DatePers;

CODANG	NUME	CNP	DataN	ADRESA	LOCALITATE	TELEFON
100	Ion Ion	1234567890100	10-JAN-1970	Mangaliei 100	Constanta	0722123456
200	Popesculon	1234567890200	10-FEB-1975	Tomis232	Constanta	0744123456
300	Ionescu Gheorghe	1234567890300	10-MAR-1980	Ferdinand 48	Mangalia	0788123456

2) Sa se selecteze toti angajatii din Constanta.

SQL> SELECT * FROM DatePers
WHERE localitate = 'Constanta ';

CODANG	NUME	CNP	DataN	ADRESA	LOCALITATE	TELEFON
100	Ion Ion	1234567890100	10-JAN-1970	Mangaliei 100	Constanta	0722123456
200	Popesculon	1234567890200	10-FEB-1975	Tomis232	Constanta	0744123456

3) Sa se afi§eze numele tuturor angajatilor care sunt din localitatile a caror nume Tncepe cu litera M.

SQL> SELECT nume, localitate
FROMDatePers WHERE
***localitate LIKE 'M%'*;**

CODANG	NUME	CNP	DataN	ADRESA	LOCALITATE	TELEFON
300	Ionescu Gheorghe	1234567890300	10-MAR-1980	Ferdinand 48	Mangalia	0788123456

4) Sa se afi§eze codul §i salariile angajatilor care au salariul de baza Tntre 6000000 §i 7000000

SQL> SELECT codang, salbaza FROMDateSal WHERE
salbaza BETWEEN 6000000 AND 7000000;

CODANG	SALBAZA
200	500000

 ISSN 2069-7988
ISSN-L 2069-7988

5) Sa se afi§eze codul angajatului cu vechime de 10 §i respectiv 15 ani

```
SQL> SELECT codang, vechime
     FROMDateSal WHERE
     vechime IN (10,15);
```

CODANG	VECHIME
100	10
300	15

6) Sa se afi§eze impozitarul in formatul in care apare in Monitorul Oficial

```
SQL> SELECT    dela \\ ' - ' \\panala ||' '|| suma \ \ ' + ' \ \
                 procent \\ ' % pentru ceea ce depaseste ' \\ dela
     FROM Impozitar;
```

DELA ||'-'|| PANALA ||"||	SUMA ||	'+' ||	PROCENT ||'%PENTRU CEEA CE DEPASESTE'		
0-2100000	0	+	18%	pentru ceea ce depaseste	0
2100001 - 5200000	378000	+	23%	pentru ceea ce depaseste 2100001	
5200001 - 8300000	1091000	+	28%	pentru ceea ce depaseste 5200001	
8300001 -11600000	1959000	+	34%	pentru ceea ce depaseste 8300001	
11600001 - 999999999999	3081000	+	40%	pentru ceea ce depaseste 11600001	

7) Sa se afi§eze, concatenat, codul angajatului §i luna din tabela Pontaj. Pentru sjrul astfel creat sa se afi§eze lungimea sa.

```
SQL> SELECTCONCAT(codang, luna), ANGAJAT LUNA
     LENGTH (concat (codang,luna))    LUNGIME SIR
     FROM Pontaj;
```

ANGAJAT LUNA	LUNGIME SIR
1001	4
2001	4
3001	4

8) Sa se afi§eze valoarea 41000/32000 rotunjita la 2 §i, respectiv 3 zecimale

```
SQL> SELECT ROUND (41000/32000, 2) 2 ZECIMALE,
                ROUND (41000/32000, 3) 3_ZECIMALE
                FROMDUAL;
```

2 ZECIMALE	3 ZECIMALE
1.28	1.281

 ISSN 2069-7988
ISSN-L 2069-7988

9) Sa se afi§eze angajatii care au cuvantul "Ion" in nume (nume §i prenume) Tmpreuna cu varsta acestora. Varsta se va afi§a in doua moduli: rotunjita in ani §i in ani cu luni.

```
SQL> SELECT nume,
        ROUND ((sysdate-datan)/365, 0)     ANI,
        ROUND ((sysdate-datan)/365,1)      ANI CU_LUNI
        FROM DatePers
         WHERE nume LIKE '%Ion% ';
```

NUME	ANI	ANI CU LUNI
Ion Ion	34	34.2
Popescu Ion	29	29.1
Ionescu Gheorghe	24	24

10) Sa se afi§eze numele angajatilor §i data Tmplinirii limitei de varsta pentru pensionare (62 de ani) precum §i numarul de luni ramase pana la pensionare (62ani*121uni).

```
SQL> SELECT nume,
        ADD MONTHS (datan, 62*12) DATA PENSIONARE
        MONTHS_BETWEEN(ADD MONTHS(datan, 62 *12),sysdate)
        LUNI PENSIONARE FROM DatePers;
```

NUME	DATA PENSIONARE	LUNI PENSIONARE
Ion Ion	10-JAN-32	334.11
Popescu Ion	10-FEB-37	395.11
Ionescu Gheorghe	10-MAR-42	456.11

11) Sa afi§eze numele angajatilor §i ultima zi a lunii corespunzatoare datei de na§tere a angajatilor din localitatea Mangalia

```
SQL> SELECT nume,
        LAST DA Y (datan) ULTIMA ZI DIN LUNA
        FROM DatePers
         WHERE localitate= 'Mangalia ';
```

NUME	ULTIMA ZI DIN LUNA
Ionescu Gheorghe	31-MAR-80

12) Sa se afis.eze urmatoarea zi de Sambata (dupa DataCurenta->sysdate)

```
SQL> SELECTNEXJDAY(sysdate,'Saturday') URMATOAREA SAMBATA
     FROMDUAL;

URMATOAREA SAMBATA 08-

OCT-05
```

13) Sa se afis.eze numele angajatilor §i data na§terii acestora Tntr-un format 'MM/YYYY' (M=Month=Luna, Y=Vear=An)

```
SQL> SELECT nume,
     TO_CHAR(datan, MM/YYYY) LUNA_AN
     FROM DatePers;
```

NUME	LUNA AN
Ion Ion	01/1970
Popescu Ion	02/1975
Ionescu Gheorghe	03/1980

14) Sa se afis.eze numele sj CNP-ul angajatilor nascuti pe 10 ianuarie 1970

```
SQL> SELECT cnp
     FROMDatePers
      WHERE datan= TO_DATE ('10 January 1970', 'ddMonth YYYY');
```

NUME	CNP
Ion Ion	1234567890100

15) Sa se afis.eze codul angajatilor, numele §i salariul acestora indexat cu 5% pentru economist §i 10% pentru director. Se stabile§te un JOIN pe tabelele DatePers §i DateSal pentru identificarea numelui §i, respectiv, codul angatilor al caror salariu va fi indexat. Salariul care nu va fi indexat va fi trecut cu SALBAZA in coloana SALINDEXAT (optiunea DEFAULT din functia DECODE).

```
SQL> SELECTds.codang, dp.nume,
     ds.salbaza SALBAZA,
     DECODE (functia, 'Ec', salbazaH.05, 'Director', salbaza*l.l, salbaza)
     SAL INDEXAT FROM DateSal ds, DatePers dp WHERE ds.codang =
     dp.codang;
```

 ISSN 2069-7988
ISSN-L 2069-7988

CODANG	NUME	SAL BAZA	SAL INDEXAT
100	Ion Ion	5000000	5250000
200	Popesculon	6500000	6500000
300	Ionescu Gheorghe	15000000	16500000

16) Sa se afis.eze suma salariilor de baza

```
SQL> SELECT 'Suma este' \\sum (salbaza)   SUMA
        FROMDatePers dp, DateSal ds WHERE
        dp.codang=ds.codang(+);
```

SUMA
Suma este 26500000

17) Sa se afis.eze numele fiecarui angajat §i codul §efului direct superior

```
SQL> SELECTname \\ 'lucreazapentru" || codsef ANGAJAT SEF
        FROMDatePers dp, DateSal ds WHERE dp. codang=ds.
        codang;
```

ANGAJAT SEF		
Ion Ion	lucreaza pentru	300
Popescu Ion	lucreaza pentru	300
Ionescu Gheorghe	lucreaza pentru	

18) Sa se afis.eze salariu de baza mediu, salariu minim §i salariu maxim pentru toti salariatii cu codul cuprins Tntre 10 §i 1000.

```
SQL> SELECTAvg (salbaza)        MEDIU,
        Min (salbaza)            MINIM,
        Max (salbaza)            MAXIM
        FROM DateSal WHERE codang BETWEEN
        10 AND 1000;
```

MEDIU	MINIM	MAXIM
7875000	3500000	16500000

19) Sa se afis.eze toti angajatii cu functia de Director, din localitatea Mangalia §i cu un salariu mai mare de 14000000.

```
SQL> SELECT nume, functia, salbaza
```

```
FROMDatePers dp, DateSal ds
WHERE
    dp. codang=ds. codang AND
    functia= 'Director'   AND dp.
    localitate= 'Mangalia' AND
    salbaza>l 4000000;
```

NUME	FUNCTIA	SALBAZA
Ionescu Gheorghe	Director	16500000

20) Sa se afi§eze toti angajatii din structura ierarhica a societatii. Radacina arborelui este Directorul.

```
SQL> SELECTLPAD (' ',5*(LEVEL-1)) \\ codang, functia
    FROM DateSal ds START WITHfunctia='Director'
    CONNECT BY PRIOR codang=codsef;
```

Rezultatul este:

LPAD(",5*(LEVEL-1))| |CODANG	FUNCTIA
300	Director
100	Ec
200	Inginer
400	Tehnician

21) Sa se blocheze randurile selectate de o cerere

```
SQL> SELECT * FROM Impozitar
    FOR UPDATE
```

Tabela blocata pentru update-area tuplurilor:

LINIE	DELA	PANALA	SUMA	PROCENT
1		0	210000	1
2	2100001	5200000	378000	23
3	5200001	8300000	1091000	28
4	8300001	11600000	1959000	34
5	11600001	99999999999	3081000	40

22) Sa se adauge un nou angajat in tabela DatePers sj sa se selecteze angajatul adaugat dupa prima litera din nume §i dupa apartenenta sa o localitate.

```
SQL> INSERT INTO DatePers VALUES
     (400, 'Popa Vasile', '1234567890400', '10-APR-1980', 'Zorelelor 12'
     , 'Medgidia', '0721333333 ');
     SELECT "from DatePers
      WHERE nume LIKE 'P%'
     AND localitate IN ('Mangalia', 'Medgidia ');
```

CODANG	NUME	CNP	DataN	ADRESA	LOCALITATE	TELEFON
400	Popa Vasile	1234567890400	10-APR-80	Zorelelorl2	Medgidia	0721333333

23) Sa se adauge datele salariale pentru angajatul nou introdus. Sa se selecteze codul, numele §i datele salariale introduse pentru noul angajat.

```
SQL> INSERT INTO DateSal
     VALUES (400, 'Tehnician',3500000,4,25,200);

     SELECT ds.codang, dp. nume, ds.functia, ds.salbaza, ds.persintr,
     ds.vechime, ds.codsef
     FROM       DatePers dp, DateSal ds
     WHERE      dp.codang=ds.codang
                AND ds.codang=400
                 OR dp.nume = '% Vasile';
```

CODANG	NUME	FUNCTIA	SALBAZA	PERSINTR	VECHIME	CODSEF
400	Popa Vasile	Tehnician	3500000	4	25	200

24) Sa se adauge in tabela Pontaj datele pentru noul angajat (cu date introduse de la tastatura).

```
SQL> PROMPT Sa se adauge in Tabela Pontaj datele pentru:
     INSERT INTO Pontaj (codang, luna, zilelucr, orezi, zileco, zilecm,
     orelucrate)
     VALUESC&CodAngajat', '&LunaPontaj', '&ZileLucr', '&OrePeZV,
     '&ZileConOdihna',' &ZileConMed',' &OreLucrEfectiv');
```

```
Sa se adauge ni Tabela Pontaj datele pentru:
Enter value for codangajat:          400
Enter value for lunapontaj:          1
Enter value for zilelucr:            22
Enter value for orepezi:             8
Enter value for zileconodihna:  1
Enter value for zileconmed:     1
Enter value for orelucrefectiv:  8
1 row created.
```

 ISSN 2069-7988
ISSN-L 2069-7988

Ulterior de poate adauga la linia de stare (o selectie explicita, prin introducerea codului corespunzator noul angajat inserat in tabela).

```
SQL> SELECT * FROM Pontaj
        WHERE codang=&CodAngajat;

Enter value for codangajat: 400

CODANG   LUNA ZILELUCR OREZI ZILECO    ZILECM    ORELUCRATE

   400      1      22             8         1            1            8
```

SAU, o selectie implicita prin specificarea directa a codului angajatului:

```
SQL> SELECT * FROM Pontaj
        WHERE codang= 400;

CODANG   LUNA ZILELUCR OREZI ZILECO    ZILECM    ORELUCRATE

   400      1      22             8         1            1            8
```

25) Sa se adauge o noua taxa, in tabela TAXE, utilizand variabile de memorie

```
SQL> ACCEPT den          PROMPT      'Denumire:'
        ACCEPT procent     PROMPT      'Procent:' ACCEPTcotamax
        PROMPT      'Cota maxima:' INSERT INTO Taxe
        VALUES('&den','&procent','&cotamax');

Denumire:          TAXA NOUA
Procent:                   2
Cota maxima:               3

Old      1: INSERT INTO Taxe VALUES          ('&den','&procent','&cotamax')
New      1: INSERT INTO Taxe VALUES          ('TAXA NOUA','2','3')
1 row created.
```

Ulterior, dupa rulare, se va putea selecta.

```
SQL> SELECT * FROM TAXE
        WHERE den = '&Denumire ';

DEN                PROCENT  COTAMAX
-------------------------------------
TAXA NOUA          2             3
```

 ISSN 2069-7988
ISSN-L 2069-7988

26) Sa se creeze o noua tabela pentru Datele Personale ale Angajatilor din Constanta (DatePersCta) §i sa se adauge ulterior in aceasta tabela datele personale ale angajatilor din Constanta existente in tabela initiala DatePers.

```
SQL> CREATE TABLE DatePersCta
        (
        codang number(5) primary key,
        nume varchar2(35),
        cnp varchar2(13),
        datan date,
        adresa varchar2(30),
        localitate varchar2(15),
        telefon varchar2(10)


        INSERT  INTO  DatePersCta
        SELECT  *  FROM  DatePers
        WHERE localitate= 'Constanta ';

        COMMIT; SELECT *

FROM DatePersCta;
```

CODANG	NUME	CNP	DataN	ADRESA	LOCALITATE	TELEFON
100	Ion Ion	1234567890100	10-JAN-1970	Mangaliei 100	Constanta	0722123456
200	Popesculon	1234567890200	10-FEB-1975	Tomis 232	Constanta	0744123456

27) Sa se majoreze salariul directorului cu 10 procente.

```
SQL> UPDATEDateSal
        SET salbaza=salbaza *1.1
        WHERE functia= 'Director';
```

Rezultatul se poate vizualiza utilizand variabila "Functia":

```
SQL> SELECT * FROM DateSal
        WHERE functia= '&Functia';

Enter value for functia: Director
```

NRCRT	CODANG	FUNCTIA	SALBAZA	PERSINTR	VECHIME	CODSEF
3	300	Director	19965000	0	15	

 ISSN 2069-7988
ISSN-L 2069-7988

28) Sa se §tearga toate lnregistrarile din DatePersCta unde numarul de telefon Tncepe cu"0744..."

SQL> DELETE FROM DatePersCta
WHERE telefon LIKE '0744%';

*SELECT * FROM DatePersCta;*

CODANG	NUME	CNP	DataN	ADRESA	LOCALITATE	TELEFON
100	Ion Ion	1234567890100	10-JAN-1970	Mangaliei 100	Constanta	0722123456

29) Sa se afis.eze numele tabelelor create in schema proprie de obiecte

SQL> SELECT tablename from USER TABLES;

TABLE NAME

DATEPERS
DATEPERSCTA
DATESAL
DEDUCERI
DEPT
EMP
IMPOZITAR
PONTAJ
SALGRADE
TAXE

30) Sa se adauge atributul TMP de tip NUMBER in tabela DatePersCta.

SQL> ALTER TABLE DatePersCta
ADD (TMPNUMBER (3));

DESCRIBE DatePersCta;

Name	Null?	lype
CODANG	NOT NULL	NUMBER(5)
NUME		VARCHAR2(35)
CNP		VARCHAR2(13)
DATAN		DATE
ADRESA		VARCHAR2(30)
LOCALITATE		VARCHAR2(15)
TELEFON		VARCHAR2(10)
TMP		NUMBER (3)

31) Sa se modifice atributul TMP la o lungime de 5 pozitii
SQL> ALTER TABLE DatePersCta

 ISSN 2069-7988
ISSN-L 2069-7988

MODIFY (TMP NUMBER (5));

DESCRIBE DatePersCta;

Name	Null?	Type
CODANG	NOT NULL	NUMBER(5)
NUME		VARCHAR2(35)
CNP		VARCHAR2(13)
DATAN		DATE
ADRESA		VARCHAR2(30)
LOCALITATE		VARCHAR2(15)
TELEFON		VARCHAR2(10)
TMP		NUMBER (5)

32) Sa se redenumeasca tabela DatePersCta in CONST

SQL> ALTER TABLE DatePersCta
RENAME TO Const;

33) Sa se §tearga tabela DatePersCta

SQL> DROP TABLE DatePersCta;

34) Sa se adauge la DatePers restrictia de Validare Codang>0.

SQL> ALTER Table DatePers
ADD (CONSTRAINT check comp CHECK (codang>0));

35) Sa se creeze tabela virtuala CONSTANTA care va contine date despre angajatii din Constanta

SQL> CREATE VIEW Constanta AS
SELECT* FROM DatePers
WHERE localitate= 'Constanta';

View created.

36) Sa se stearga tabela virtuala CONSTANTA

SQL> DROP VIEW Constanta;

View dropped.

ISSN-L 2069-7988

37) Sa se afi§eze numaml de Tnregistrari din tabela DatePers

```
SQL> SELECT count (*) NR INREG
     FROM DatePers;
```

NR INREG

38) Sa se vizualizeze restrictiile tabelei DatePers

```
SQL> SELECT
          CONSTRAINT TYPE,
          CONSTRAINT NAME,
          STATUS
     FROM USER CONSTRAINTS WHERE
     TABLE NAME= 'DATEPERS';
```

C	CONSTRAINT NAME	STATUS
C	CCHECK COMP	ENABLED

7.2. Aplicatie informatica pentru activitatea de aprovizionare si desfacere a uneifirme

1. Crearea bazei de date.
1) Sa se creeze tabelele clienti, furnizori, produse, tranzactii, documente §i proddoc.

CREATE TABLE clienti

```
code    varchar2 (5).
dene    varchar2 (30),
adr     varchar2 (30),
loc     varchar2 (20),
cont    varchar2 (11),
banca   varchar2 (15),
    constraint pkcodc primary key (code)
```

CREATE TABLE furnizori

```
(
   codf     varchar2 (5),
   denf     varchar2 (30),
   adr      varchar2 (30),
   loc      varchar2 (20),
   cont     varchar2 (11),
   banca    varchar2 (15),
     constraint pkcodfprimary key (codf)
```

CREATE TABLE produse

```
codp    varchar2 (5),
denp    varchar2 (25),
um      varchar2 (5),
pret    number (10),
stoc    number (5),
termen  date,
    constraint pkcodp primary key (codp)
```

```
CREATE TABLE tranzactii
(
    codt        varchar2 (5),
    dent        varchar2 (1)
        constraint nndent not null
        constraint ckdent check (upper (dent) in ('L', 'R')),
    dataora     date default sysdate,
    codf        varchar2 (5),
    code        varchar2 (5),
        constraint pkcodt primary key (codt),
        constraint fkcodfforeign key (codf) references furnizori (codf),
        constraint fkeode foreign key (code) references clienti (code)


CREATE TABLE documente
(
    codd        number (5)
        constraint ckcodd check (codd>0),
    dend        varchar2 (4)
        constraint nndend not null
        constraint ckdend check (upper (dend) in
    (FACT, A VIZ[1], 'MR', 'CHIT')),
    data        date default sysdate,
    codt        varchar2 (5),
        constraint pkcodd primary key (codd),
        constraint fkcodt foreign key (codt) references tranzactii (codt)


CREATE TABLEproddoc

codd        number(5),
codp        varchar2(5),
um          varchar2(5),
cant         number(5),
        constraint pkcoddp primary key (codd,codp)
```

2. Modificarea structurii tabelelor bazei de date

1) Sa se modifice dimensiunea atributului CodP din tabela Produse, la 4 caractere.

SQL> PROMPT Modificati dimensiunea atributului Codp din tabela Produse la 4 caractere
SQL> ALTER TABLE produse MODIFY (codp varchar2 (4));
SQL> DESCRIBE produse;

Name	Null?	Type
CODP	NOT NULL	VARCHAR2 (4)
DENP		VARCHAR2 (40)
UM		VARCHAR2 (5)
PRET		NUMBER (13)
STOC		NUMBER (7)
TERMEN		DATE

2) Adaugati atnbutul IE (number(2)) tabelei ProdDoc

SQL> PROMPT Adaugati atributul IE (number (2)) tabelei proddoc
SQL>ALTER TABLE proddoc ADD (IE NUMBER (2)); SQL> DESCRIBE proddoc;

Name	Null?	Type
CODD	NOT NULL	NUMBER (5)
CODP	NOT NULL	VARCHAR2 (5)
UM		VARCHAR2 (5)
CANT		NUMBER (6)
IE		NUMBER (2)

3) Adaugati atributul Valoare, numeric de 20 caractere, la tabela Documente.

SQL> PROMPT Adaugati atributul valoare la tabela documente
SQL> ALTER TABLE documente ADD (valoare number (20));
SQL> DESCRIBE documente;

Name	Null?	Type
CODD	NOT NULL	NUMBER (5)
DEND	NOT NULL	VARCHAR2 (4)
DATA		DATE
CODT		VARCHAR2 (5)
VALOARE		NUMBER (20)

3.1nserare inregistrari in tabele.
SQL> DELETE FROM clienti;
SQL> DELETE FROMfurnizori;

```
SQL> DELETE FROMproduse;
SQL> DELETE FROM tranzactii;
SQL> DELETE FROM documente;
SQL> DELETE FROMproddoc;

SQL> PROMPT INSERARE IN TABELA CLIENTI;
SQL> INSERT INTO clienti VALUES ('1', 'GOODS ', 'PIPERA
     135', 'BUCURESTI', Al 234567890', 'BRD'); SQL> INSERT
INTO clienti VALUES ('2', 'DepozitPC, 'Stefan eel
     Mare 110', 'Bucuresti', Al231231234', 'BCR'); SQL>
INSERT INTO clienti VALUES ('3', 'Flamingo', 'Mihai
     Eminescu 18', 'Cluj', A1231231235', 'BCR'); SQL> INSERT
INTO clienti VALUES ('4', 'Ultra Pro', 'Mihai Bravu
     11', 'Timisoara', 'Bl 231231234', 'BRD '); SQL> INSERT INTO
clienti VALUES ('5', 'Flanco', 'Dorobantilor
     130', 'Cluj', 'Cl 231231234', 'BCR');

SQL> PROMPT INSERARE IN TABELA FURNIZORI;
SQL> INSERT INTO furnizori VALUES ('1','GOODS','PIPERA
     135', 'BUCURESTI', Al 2345 67890', 'BRD'); SQL> INSERT
INTO furnizori VALUES ('2','ComputerNT','Gral
     Popescu 13', 'Iasi', Al 234123412', 'BRD '); SQL> INSERT
INTO furnizori VALUES ('3', 'Python', 'Charles de
     Gaule 117', 'Cluj', Al 234512345', 'BCR'); SQL> INSERT
INTO furnizori VALUES ('4','Blue Ridge','Magheru
     307', 'Bucuresti', 'B12345 54321', 'BRD'); SQL>
INSERT INTO furnizori VALUES ('5','Deck
     Electronics', LaculAlb 35', 'Iasi', 'Bl234567777', 'BCR');

SQL> PROMPT INSERARE IN TABELA PRODUSE;
SQL> INSERT INTO produse VALUES('PV,'Monitor
     7inch', 'buc',3 500000,1000,
     TO_DA TE('01/08/2006', 'DD/MM/YYYY')); SQL>
INSERT INTO produse VALUES('P2','CD-RW ASUS
     24xl0x40x', 'buc',1000000,500,
     TO_DA TE('01/08/2005', 'DD/MM/YYYY'));
SQL> INSERT INTO produse VALUES('P3','Tastatura
     qwerty', 'buc ',300000,100,
     TO_DA TE('01/06/2004', 'DD/MM/YYYY'));
```

```
SQL> INSERT INTO produse VAL UES('P4', 'CPU AMD Athlon
     1.4GHz', 'buc ',2700000,700, TO_DA TE('01/12/2004',
     'DD/MMfYYYY'));
SQL> INSERT INTO produse VALUES('P5', 'Mouse
     A4TECH', 'buc',100000,150, TO_DA TE('01/06/2
     004', 'DD/MM/YYYY'));

SQL> PROMPTINSERARE IN TABELA TRANZACTII;
SQL> INSERT INTO tranzactii VALUES
     ('Tl', 'R', TO_DA TE('01/08/2003 02:12:39', 'MM/DD/YYYY
     HH:MI:SS'),'3','V); SQL> INSERT
INTO tranzactii VALUES
     ('T2', 'R',TO  DATE('11/10/2003 10:20:09', 'MM/DD/YYYY
     HH:MI:SS'),'4','V); SQL> INSERT
INTO tranzactii VALUES
     ('T3', 'L', TO_DA TE('12/10/2003 12:12:30', 'MM/DD/YYYY

SQL> INSERT INTO tranzactii VALUES
     ('T4', 'L',TO_DATE('02/11/2003 04:55:39', 'MM/DD/YYYY


SQL> PROMPT INSERARE IN TABELA DOCUMENTE;
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (10123, 'IACT, TO_DATE('01/08/2003', 'MM/DD/YYYY'), 'Tl');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (20123, 'MR', TO_DATE( '01/08/2003', 'MM/DD/YYYY'), 'Tl ');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (10124, 'FACT',TO_DATE('l 1/10/2003', 'MM/DD/YYYY'), 'T2');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (20124, 'MR', TO_DATE('l 1/10/2003', 'MM/DD/YYYY'), 'T2 ');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (30122, A VIZ', TO_DATE('12/10/2003', 'MM/DD/YYYY'), 'T3');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (10125, 'FACT',TO  DATE('12/10/2003', 'MM/DD/YYYY'), 'T3');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (30123, A VIZ', TO_DATE('02/11/2003', 'MM/DD/YYYY'), 'T4');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (10126, 'FACT,TO_DATE('02/11/2003', 'MM/DD/YYYY'), 'T4');
SQL> INSERT INTO documente (codd,dend,data,codt) VALUES
     (40123,' CHIT,TO_DATE('02/11/2003', 'MM/DD/YYYY'), 'T4');
```

Valorile pentru campul valoare din tabela Documente nu au fost direct introduse in tabela, deoarece acest camp este unul calculat, iar valorile sale se vor introduce printr-o formula.

```
SQL> PROMPT INSERARE IN TABELA PRODDOC; SQL>
INSERT INTO proddoc VALUES (10123, Tl','buc',500,null); SQL>
INSERT INTO proddoc VALUES (10123,'P2','buc',500,null); SQL>
INSERT INTO proddoc VALUES (20123,'PV,'buc',500,null); SQL>
INSERT INTO proddoc VALUES (20123,'P2','buc',500,null); SQL>
INSERT INTO proddoc VALUES (10124,'P3','buc',100,null); SQL>
INSERT INTO proddoc VALUES (10124,'P4','buc',500,null); SQL>
INSERT INTO proddoc VALUES (10124,'P5','buc',100,null); SQL>
INSERT INTO proddoc VALUES (20124,'P3','buc',100,null); SQL>
INSERT INTO proddoc VALUES (20124,'P4','buc',450,null); SQL>
INSERT INTO proddoc VALUES (20124,'P5','buc',100,null); SQL>
INSERT INTO proddoc VALUES (30122,'PV,'buc',100,null); SQL>
INSERT INTO proddoc VALUES (30122,'P2','buc',200,null); SQL>
INSERT INTO proddoc VALUES (10125,'PV,'buc',100,null); SQL>
INSERT INTO proddoc VALUES (10125,'P2','buc',200,null); SQL>
INSERT INTO proddoc VALUES (30123,'PV,'buc',300,null); SQL>
INSERT INTO proddoc VALUES (30123, 'P4','buc',500,null); SQL>
INSERT INTO proddoc VALUES (10126,'Pl','buc',300,null); SQL>
INSERT INTO proddoc VALUES (10126,'P4','buc',500,null);
```

4. Definirea generatorului de numere de secventa:
1) Sa se creeze o secventa SECV care Tncepe cu valoarea 10127 §i se termina cu valoarea 10130 §i pasul 1. Acesta secventa secv se va folosi ulterior pentru generarea automata de numere unice pentru campul codd din tabela Documente. Se vor genera succesiv, crescator, numerele cuprinse Tntre 10127 §i 10130.

```
SQL>CREATE SEQUENCE secv   //nume secventa
      INCREMENT BY 1       //pasul de incremental
      START WITH 10127     //valoarea depornire a secventei
      MAXVALUE 10130       //valoarea maxima a secventei
      NOCACHE NOCYCLE;     //secventa finita
```

2) Sa se adauge o noua valoare pentru atributul cheii primare codd din tabela Documente folosindu-se succesiunea generata de secventa SECV anterior creata.

SQL> INSERT INTO documente VALUES (secv.nextval, 'FACT', sysdate-2, 'T5', null); SQL> SELECT "from documente;

CODD	DEND	DATA	CODT	VALOARE
10123	FACT	08-JAN-05	Tl	
20123	NIR	08-JAN-05	Tl	
10124	FACT	10-NOV-05	T2	
20124	NIR	10-NOV-05	T2	
30122	AVIZ	10-DEC-05	T3	
10125	FACT	10-DEC-05	T3	
30123	AVIZ	ll-FEB-05	T4	
10126	FACT	ll-FEB-05	T4	
40123	CHIT	ll-FEB-05	T4	
10127	FACT	26-FEB-06	T5	

La executie se observa adaugarea tuplului 10127-FACT-26FEB04-T5, cheia primara astfel defmita, pentru campul codd, frind prima valoare a secventei SECV. La fiecare apelare a cuplului INSERT-SELECT secventa SECV anterior creata va incrementa automat cheia primara Codd din tabela Documente.

3) Sa se adauge Tnregistrarile corespunzatoare pentru o receptia a 100 de bucati din produsul P3 §i alte 200 de bucati din produsul P4 de la furnizorul 4, §tiindu-se ca factura a fost emisa de furnizor cu 2 zile Tnainte de receptia produselor.

SQL> LNSERTLNTO tranzactii VALUES ('T5','R', sysdate, '4','1'); SQL> LNSERTLNTO documente VALUES (secv.nextval, 'FACT, sysdate-2, 'T5',0);
*SQL> LNSERTLNTO documente VALUES (20125, 'NLR', sysdate, 'T5', 0); SQL> LNSERTLNTOproddoc VALUES (secv. currval, 'P3', 'buc ',100,0); **
*SQL> LNSERT LNTO proddoc VALUES (secv. currval, 'P4', 'buc ',200,0); **
SQL> LNSERT LNTO proddoc VALUES (20125,'P3','buc',100,l);
SQL> LNSERT LNTO proddoc VALUES (20125, 'P4', 'buc', 200,1);

Pentm a se putea defini cerinta ca factura sa fie emisa cu doua zile Tnainte de receptia produselor, s-a optat pentru varianta sysdate-2 (data curenta-doua zile), Tntrucat receptia produselor intrate in gestiune se face in ziua curenta de lucru. Cele doua tupluri sunt cele care defmesc aprovizionarea produselor P3 §i P4, avand drept valori, pentru unul din cele doua atribute ale cheii primare, codd - numarul de secventa curent (secv.currval) definit anterior §i preluat de la generatorul secv.nextval (din Tnregistrarea: secv.nextval - FACT- sysdate-2, T5, 0)

Adaugarile la stoc ale celor doua produse se vor regasi in tabela Proddoc.

inainte de inserare:
*SQL> select * from documente;*

CODD	DEND	DATA	CODT
20123	NIR	08-JAN-05	Tl
10124	FACT	10-NOV-05	T2
20124	NIR	10-NOV-05	T2
30122	AVIZ	10-DEC-05	T3
10125	FACT	10-DEC-05	T3
30123	AVIZ	ll-FEB-05	T4
10126	FACT	ll-FEB-05	T4
40123	CHIT	ll-FEB-05	T4
10123	FACT	08-JAN-05	Tl

Dupa inserare:
*SQL> select * from documente;*

CODD	DEND	DATA	CODT	VAL
20123	NIR	08-JAN-05	Tl	
10124	FACT	10-NOV-05	T2	
20124	NIR	10-NOV-05	T2	
30122	AVIZ	10-DEC-05	T3	
10125	FACT	10-DEC-05	T3	
30123	AVIZ	ll-FEB-05	T4	
10126	FACT	ll-FEB-05	T4	
40123	CHIT	ll-FEB-05	T4	
10123	FACT	08-JAN-05	Tl	
10127	FACT	27-FEB-06	T5	0
20125	NIR	29-FEB-06	T5	0

Campurile adaugate in cele doua tabele §i avand valoarea cheii primare generata ca fiind 10127 (primul numar din secventa SECV) sunt cele corespunzatoare instructiunilor SECV.NEXTVAL §i SECV.CURRVAL, care au generat, respectiv, preluat valorile pentru cheia primara.

4) Sa se afis.eze ultimul numar utilizat din secventa SECV:

SQL> SELECTsecv.CURRVAL
FROMDUAL;

CURRVAL

10127

5) Sa se modifice pasul secventei SECV de la 1 la 10000

SQL> ALTER SEQUENCE secv
INCREMENT BY 10000;
Sequence altered.

6) Sa se §terga secventa SECV .

SQL> DROP SEQUENCE secv;
Sequence dropped.

5. Actualizari la nivelul aplicatiei:

1) Sa se insereze in atributul IE din tabela Proddoc, valorile: "l",pentru NIR (I) ; ,,-1", pentru AVIZE (E); "0", pentru celelalte documente.

SQL> UPDATE proddoc SETIE= -1
WHERE SUBSTR (TOjCHAR (codd), 1,1)= '3 ';
SQL> UPDATE proddoc SET IE=1
WHERE SUBSTR (TOjCHAR (codd), 1,1)= '2 ';
SQL> UPDATE proddoc SET IE=0
WHERE SUBSTR (TOjCHAR (codd), 1,1) N0TIN('2 ', ' 3 ');
SQL> SELECT FROM proddoc;*

CODD	CODP	UM	CANT	IE
10123	P2	buc	500	0
20123	PI	buc	500	1
10123	PI	buc	500	0
20123	P2	buc	500	1
30123	PI	buc	300	-
30123	P4	buc	500	-
10126	PI	buc	300	0
10126	P4	buc	500	0
10127	P3	buc	100	0
10127	P4	buc	200	0
20125	P3	buc	100	1
20125	P4	buc	200	1

2) Sa se calculeze §i sa se afi§eze valoarea totala pentru fiecare document, din tabela Documente.

```
SQL> UPDA TE documente D SET
        valoare =
        (SELECT SUM (cant*pret) valoare
        FROM proddoc PD, produse P
         WHERE Pcodp=PD.codp AND D.codd=PD.codd
         GROUP BY D.codd);
SQL> SELECT codd, valoare
         FROM documente
         WHERE valoare IS NOT NULL;
```

CODD	VALOARE
20123	2.250E+09
10124	1.390E+09
20124	1.255E+09
30122	550000000
10125	550000000
30123	2.400E+09
10126	2.400E+09
10127	570000000
20125	570000000
10123	2.250E+09

3) Sa se diminueze stocul aferent prodului P5 cu 50 de bucati.

```
SQL> UPDATE produse
                SET stoc= stoc - 50
                 WHERE codp='P5'; SQL> SELECT
codp, denp, stoc from produse;
```

CODP	DENP	STOC
P5	Mouse A4TECH	150

6. Functiile pentru §iruri de caractere:

1) Sa se selecteze numele §i localitea unde T§i au sediul clientii, folosind formatul de afi§are cu prima litera majuscula.

```
SQL> SELECT
        INITCAP (DENC) LITERA MARE NUME,
        INITCAP(LOC)
```

FROM clienti;

LITERA	MARE NUME	INITCAP (LOC)
Intercon		Bucuresti
Depozit	De Calculatoare	Bucuresti
Flaming		Cluj
Ultra		Timisoara
Flanco		Cluj

2) Sa se concateneze s.irurile corespunzatoare atributelor ,,Adresa" §i ,,Localitate" din tabela furnizori, pentru furnizorul "Blue Ridge" .

SQL> SELECT denf, CONCAT (adr, loc) "AdesadinLocalitatea"
FROM Furnizori WHERE denf = 'Blue Ridge ';

DENF	Adresa _ din _ Localitatea
Blue Ridge	Magheru 307 Bucuresti

3) Sa se selecteze toti furnizorii, aducand coddf, denf §i loc la lungimea de 20 de caractere fiecare, utilizand LPAD §i RPAD.

*SQL> SELECT LPAD (codf 20, ' * '),*
LPAD (denf, 20), LPAD (loc, 20, '-')
FROM furnizori;

LPAD(CODF,20,'*')		LPAD(DENF,20)	LPAD(LOC,20,'-')
**********	*1*	INTERCONN	----- Bucuresti
**********	*2*	Computer Network	--------Iasi
**********	3	Python Blue Ridge	--------Cluj
**********	4	Deck Electronics	--------Bucuresti
**********			--------Iasi

*SQL> SELECT RPAD (codf, 20, ' * '),*
RPAD (denf, 20),
RPAD (loc, 20, '-')
FROM furnizori ;

RPAD(CODF,20,'')*	*RPAD(DENF,20)*	*RPAD(LOC,20,'-')*
1 **********	*INTERCONN*	*BUCURESTI—*
2 **********	*Computer Network*	*Iasi----------*
3 **********	*Python*	*Cluj ----------*
4 **********	*Blue Ridge*	*Bucuresti ---------*
5 **********	*Deck Electronics*	*Iasi----------*

4) Sa se afi§eze furnizorii din alte localitati decatBucures.ti.

```
SQL> SELECT codf, denf, loc FROMfurnizori
        WHERE UPPER (loc) <> 'BUCURESTF;
```

CODF	DENF	LOC
2	Computer Network	Iasi
3	Python	Cluj
5	Deck Electronics	Iasi

5) Sa se afi§eze clientii a caror denumire Tncepe cu litera "F"

```
SQL> SELECT code, dene FROM clienti
        WHERE SUBSTR (denc,l,l)= 'F';
```

CODC	DENC
3	Flamingo
5	Flanco

9. Functiile de data

1) Sa se afi§eze denumirea furnizorilor cu care nu s-au mai Tncheiat tranzactii in ultimele 6 luni.

```
SQL> SELECT codf, denf FROMfurnizori
        WHERE codf NOT IN
                (
                SELECT codf FROM tranzactii
                 WHERE MONTHS_BETWEEN (sysdate,dataora)<=6
```

CODF	DENF
2	Computer Network
3	Python
5	Deck Electronics

2) Sa se afi§eze perioada (lunile) de garantie ramase pana la expirarea produselor (inventariate in tabela Produse) cu enumerarea doar a celor care mai au ca valabilitate minimum 3 luni.

```
SQL> SELECT codp, denp,
```

MONTHS_BETWEEN (termen, sysdate) LUNI GARANTIE
FROM produse
where MONTHS_BETWEEN (termen, sysdate) >3;

CODP	DENP	LUNI GARANTIE
PI	Monitor 17inch	29.0727
P2	CD-RW ASUS 24xl0x40x	17.0727
P3	Tastatura qwerty	3.0727001
P4	CPU AMD AtMon 1.4GHz	9.0727001
P5	Mouse A4TECH	3.0727001

3) Sa se selecteze produsul cu termenul de garantie eel mai Tndepartat (August 2007) §i sa se evidentieze lunile de garantie ramase de la data curenta la termen.

SQL> SELECT codp, denp,
MONTHSBETWEEN ('Ol-Aug-06', sysdate)
L UNI_MAXIME_ GARANTIE
FROM produse
WHERE termen='01-Aug-07';

CODP	DENP	LUNI MAXIME GARANTIE
PI	Monitor 17 inch	29.072489

4) Sa se afis.eze, codul, denumirea, termenul de garantie, precum §i data decalata cu trei luni fata de termenul de garantie §i data anetrioara cu trei luni termenului de garantie. Se vor evidentia produsele ale caror termene de valabilitate nu au expirat.

SQL> SELECT codp, denp, termen,
ADD MONTHS (termen, 3) PESTE TREI L, ADD
MONTHS (termen, -3) CUJTREI L IN URMA
FROM produse WHERE termen>sysdate;

CODP	DENP	TERMEN	PESTE TREI L	CU TREI L IN *URMA*
PI	Monitor 17inch	01-AUG-07	01-NOV-07	01-MAY-07
P2	CD-RW AS	01-AUG-06	01-NOV-06	01-MAY-06
P3	Tastatura	01- JUN-05	01- SEP- 05	01-MAR-05
P4	CPU AMD 1.4GHz	01- DEC-05	01-MAR-06	01-SEP- 05
P5	Mouse A4TECH	01- JUN-05	01- SEP- 05	01-MAR-05

5) Sa se afi§eze data urmatoarei zile a saptamanii *(char)* dupa o data declarata.

SQL> SELECT NEXT DAY('01-MAR-05', 1)
FROM dual;

NEXT DAY 06-

MAR-05

SQL> SELECT NEXT DAY ('01-MAR-05', 2)
FROM dual;

NEXT DAY 07-

MAR-05

5) Sa se afi§eze ultima zi a lunii *(char)* dupa o data declarata.

SQL> SELECT LAST DAY ('01-jun-05')
FROM dual;

LAST DAY 30-

JUN-05

SQL> SELECT codp, denp, termen,
LAST DAY (termen) ULTIMA ZI LUNA
FROM produse;

CODP	DENP	TERMEN	ULTIMA ZI LUNA
PI	Monitor 17inch	01-AUG-07	31- AUG-07
P2	CD-RW ASUS 24xl0x40x	01-AUG-06	31 -AUG-06
P3	Tastatura qwerty	01 -JUN-05	30 - JUN-05
P4	CPU AMD AtMon 1.4GHz	01 -DEC-05	31 - DEC-05
P5	Mouse A4TECH	01 -JUN-05	30 -JUN-05.

Functia *ROUND* poate fi aplicata pe date calendaristice. *Round (dataJ)intoarce datal* cu timpul setat la 12:00AM (noaptea). Aceasta este folositoare atunci cand se compara date care au timpuri diferite.

ROUND (datal, 'MONTH') Tntoarce:

- prima zi a lunii continand *datal,* daca *datal* este in prima parte a lunii,
- prima zi a urmatoarei luni, daca *datal* este in a doua jumatate a lunii

- *ROUND(data 1,' YEAR')* Tntoarce:
- prima zi a anului continand *datal,* daca *datal* este in prima jumatate a anului,
- prima zi a urmatorului an, daca *datal* este in a doua jumatate a lunii

De exemplu:
6) Sa se foloseasca functia ROUND pentru a returna prima zi a lunii sau anului sau prima zi a urmatoarei luni sau an, in functie de data declarata.

```
SQL> SELECTSYSDATE,
     ROUND (SYSDATE, 'MONTH') LUNA ROTUNJITA,
     ROUND (SYSDATE,'YEAR') ANULROTUNJIT
     FROMDUAL;
```

SYSDATE	LUNA ROTUNJITA	ANUL ROTUNJIT
02-SEP-05	01-SEP-05	01-JAN-06

7) Analog, sa se identifice rezultatele Tntoarse de functia LASTDAY pentru valorile atributului TERMEN din tabela Produse:

```
SQL> SELECT codp, denp, termen,
     LAST DAY(termen) ULTIMA ZI LUNA
     FROM produse;
```

CODP	DENP	TERMEN	ULTIMA ZI LUNA
PI	Monitor 17inch	01-AUG-07	31-AUG-07
P2	CD-RW ASUS 24xl0x40x	01-AUG-06	31-AUG-06
P3	Tastatura qwerty	01-JUN-05	30-JUN-05
P4	CPU AMD Athlon 1.4GHz	01-DEC-05	31-DEC-05
P5	Mouse A4TECH	01-JUN-05	30-JUN-05

8) Functia *TRUNC(datal,'char')* gases.te prima zi a lunii care e continuta in *datal,* daca *char = 'MONTH' sau* gase§te prima zi a anului care contine *dataldaca char= 'YEAR'.* Sa se utilizeze facilitatile acestei functii.

```
SQL> SELECT SYSDATE DATA CURENTA,
     TRUNC (SYSDATE, 'MONTH') PRIMA ZI LUNA,
     TRUNC (SYSDATE, 'YEAR') PRIMA ZI AN FROM
     SYSDUAL;
```

DATA CURENTA	PRIMA ZI LUNA	PRIMA ZI AN
02-OCT-05	01-OCT-05	01-JAN-05

8. Functii matematice:

1) Sa se afi§eze lungimea atributului Denumire client din tabela Clienti

```
SQL> SELECT dene,
        LENGTH (dene) LUNGIME NUME
        FROM clienti;
```

DENC	LUNGIME NUME
INTERCONN	9
Depozitul de calculatoare	25
Flamingo	8
Ultra Pro	9
Flanco	6

2) Sa se afi§eze comisionul corespunzator vanzarii fiecarui produs, in mii lei

```
SQL> ACCEPT       comision PROMPT  'Introduced comision:  ';
SQL> SELECT       denp,pret,
        & comision COMISION (%) pret*&comision/1000
        VALOARE COMISION FROMproduse;
```

Introduced comision: 10

DENP	PRET	COMISION (%)	VALOARE COMISION
Monitor 17inch	3500000	10	35000
CD-RW ASUS 24xl0x40x	1000000	10	10000
Tastatura qwerty	300000	10	3000
CPU AMD Athlon 1.4GHz	2700000	10	27000
Mouse A4TECH	100000	10	1000

3) Sa se calculze §i afi§eze stocul initial pentru fiecare produs in parte.

```
SQL> SELECT       Pcodp,
        stoc STOCJNITIAL,
        SUM (stoc+IE*cant) STOC  CURENT
        FROM      produse P, proddoc PD
```

 ISSN 2069-7988
ISSN-L 2069-7988

```
        WHERE       P.codp = PD.codp
        GROUPBY     P.codp, stoc;
```

CODP	STOC INITIAL	STOCCURENT
PI	1000	6100
P2	500	2300
P3	100	600
P4	700	4350
P5	100	300

4) Sa se afis.eze denumirea, pretul §i stocul actual al produselor, sub forma: PRODUSUL «nume» ARE PRETUL UNITAR: «pret» LEI. STOCUL ACTUAL ESTE: «stoc» «um»

```
SQL> SELECT 'PRODUSUL ' \\ LOWER (denp) ||
        'AREPRETUL UNITAR: ' \\pret\\ LEI.
        STOCUL ACTUAL ESTE: ' \\stoc\\ 'DE'\\um
        FROM produse;
```

PRODUSUL Monitor 17inch ARE PRETUL UNITAR: 3500000 LEI. STOCUL ACTUAL ESTE: 1000 DE buc
PRODUSUL Cd-rw asus 24xl0x40x ARE PRETUL UNITAR: 1000000 LEI. STOCUL ACTUAL ESTE: 500 DE buc
PRODUSUL Tastatura qwerty ARE PRETUL UNITAR: 300000 LEI. STOCUL ACTUAL ESTE: 100 DE buc
PRODUSUL CPU amd athlon 1.4ghz ARE PRETUL UNITAR: 2700000 LEI. STOCUL ACTUAL ESTE: 700 DE buc
PRODUSUL Mouse a4tech ARE PRETUL UNITAR: 100000 LEI. STOCUL ACTUAL ESTE: 100 DE buc

5) Sa se afis.eze codul produsului §i pretul marit cu 1.1 pentru Monitoare §i cu 1.2 pentru Mouse.

```
SQL> SELECT codp,
        pret PRET INITIAL,
        DECODE      (denp, 'monitor 17inch', pret*l.l,
                    'mouse A4TECH', pret*1.2, pret) PRET MARIT
        FROM produse;
```

CODP	PRET INITIAL	PRET MARIT
PI	3500000	3850000
P2	1000000	1000000
P3	300000	300000
P4	2700000	2700000
P5	100000	120000

6) Sa se afieze Cantitatea Medie cumparata din fiecare produs §i sa se ordoneze tuplurile dupa Cantitate.

```
SQL> SELECT P.codp, AVG (cant) CANT ME DIE
     FROMproduse P, proddoc PR WHERE
     Pcodp= PRcodp AND IE =1 GROUP BY
     P.codp ORDER BY A VG (cant);
```

COD P	CANT MEDIE
P3	100
P5	100
P4	325
PI	500
P2	500

Ca algoritm de analiza al functiei DECODE se poate observa ca, pentru coloana DENP (primul argument) are loc cautarea valorilor "monitor 17 inch" §i "mouse A4 Tech", iar in cazul in care acestea sunt regasite pe coloana denumirilor, preturile lor sunt actualizate cu 1.1 §i, repsectiv, 1.2.

Pentru restul produselor care nu fac obiectul cautarii, se trece implicit, ultimul argument, in cazul de fata coloana PRET, sau se poate trece o expresie 'PretNemodificat'

Fiind vorba de cumparare, implicit se ia in calcul ca document de intrare NIR-ul (pentru aprovizionare), acest lucru necesitand o conditie suplimentara IE=1 (alaturi de cea care identified din tabela Produse doar acele produse care au facut obiectul tranzactiei §i au la baza un document justificativ).

7) Sa se afi§eze doar acele produse care au cantitatea minima vanduta mai mare decat cantitatea minima a produsului P3.

```
SQL>
SELECT codp,
       MIN(cant) CANT MINIMA
       FROM proddoc WHERE IE= -1
       GROUP BY codp HA VING MIN
       (cant) >
              (SELECT MIN (cant)
                     FROMproddoc
                     WHERE codp='P3');
```

CODP	CANT MINIMA
P2	200
P4	500

Fiind vorba de vanzare se porne§te de la ideea ca documentul justificativ aferent iesjiii din gestiune este avizul, ca atare, se va trece conditia IE=-1. Totodata, in aceasta situatie este vorba de o clauza select imbricata pentru a permite selectia doar a celor produse care respecta o conditie fata de produsul P3. Ca §i in cazul anterior nu se va trece in clauza select atributul *cant* dupa pentru care se calculeaza functiile §i se face gruparea.

8) Sa se afis.eze cantitatea medie doar pentru produsele care apar mai mult de doua ori in tab el a Proddoc.

```
SQL> SELECT codp,
        AVG (cant) CANT MEDIE
        FROMproddoc
        GROUP BY codp
        HA VING COUNT (*) > 2;
```

CODP	CANT MEDIE
PI	300
P2	350
P3	100
P4	391.66667

9) Sa se afis.eze doar acele produse pentru care cantitatea este mai mare sau egala cu 200.

```
SQL> SELECT codp, MAX (cant) CANT MAXIMA
        FROMproddoc HA VING MAX (cant) > = 200
        GROUP BY codp;
```

CODP	CANT MAXIMA
PI	500
P2	500
P4	500

10) Sa se afi§eze doar acele produse pentru care cantitatea medie este mai mare sau egala cu 200.

```
SQL> SELECT codp, A VG (cant) MEDIE
        FROMproddoc
```

```
GROUP BY codp
HA VING A VG (cant) > 200;
```

CODP	MEDIE
PI	300
P2	350
P4	391.66667

11) Sa se afis.eze cantitatea medie pe tip de produs, pentru toate codurile de produs mai putin PI.

```
SQL> SELECT codp, A VG (cant) MEDIE CANT
     FROMproddoc WHERE codp != TV
     GROUP BY codp;
```

CODP	MEDIE CANT
P2	350
P3	100
P4	391.66667
P5	100

12) Sa se calculeze cantitatea medie pentru fiecare produs distinct, din tab el a Proddoc.

```
SQL> SELECT codp,
     AVG (cant) MEDIE
     FROMproddoc
     GROUP BY codp;
```

CODP	MEDIE
PI	300
P2	350
P3	100
P4	391.66667
P5	100

13) Determinati pretul mediu pentru fiecare produs in afara de produsul 'Monitor 17inch' din tabela Produse.

```
SQL> SELECT codp,
     AVG (pret) PRET MEDIU
     FROM produse
      WHERE denp!= 'Monitor 17inch'
```

```
GROUP BY codp;
```

CODP	PRET MEDIU
P2	1000000
P3	300000
P4	2700000
P5	100000

14) Afi§ati pretul minim pe produs.

```
SQL> SELECT denp,
     MIN (pret) PRET MINIM
     FROMproduse GROUP BY
     denp;
```

DENP	PRET MINIM
CD-RW ASUS 24xl0x40x	1000000
CPU AMD Athlon 1.4GHz	2700000
Monitor 17inch	3500000
Mouse A4TECH	100000
Tastatura qwerty	300000

15) Sa se afi§eze toate produsele cu diferente cantitative in documente.

```
SQL> SELECT a.codp
          FR0M(
               SELECTp. codp, SUM (cant) cant
                    FROMproddoc p,documente d
                    WHERE p. codd=d codd
                         AND dend='FACT'
                    GROUP BY p.codp
               )a,
               (SELECTp.codp, SUM(cant) cant
                    FROM proddoc p,documente d
                    WHERE p. codd=d codd
                         AND dendo'FACT
                    GROUP BY p.codp
               )b  WHERE
     a.codp=b.codp
          AND a.cant-b.cant <> 0;
```

CODP P4

Se identifica cu documentele FACTURA care se regasesc atat in nomenclatorul de documente (tabela Documente) cat §i in nomenclatorul de documente "tranzactionate" (participante la tranzactiile de produse, din tabela Proddoc). Apoi se identifica cu restul de documente (in afara de FACTURA) aflate atat in nomenclator, cat §i in tranzactii. in final se tree conditiile de identificare a produselor tranzactionate §i se stabilesc diferentele cantitative.

16) Sa se afi§eze denumirea, pretul §i valoarea totala a vanzarilor pentru fiecare produs, tinand cont de comisionul de 5%.

```
SQL> SELECT denp, pret,
        SUM(pret*cant*1.05) TOTAL VANZARI
        FROM produse, proddoc
         WHERE produse. codp=proddoc.codp
                  AND IE= -1
        GROUP BY denp,pret;
```

DENP	PRET	TOTAL VANZARI
CD-RW ASUS 24xl0x40x	1000000	210000000
CPU AMD Athlon 1.4GHz	2700000	1.418E+09
Monitor 17inch	3500000	1.470E+09

S-au identificat doar produsele pentru care IE=-1, respectiv au ie§it din gestiune (au fost vandute) fiind Tnsotite de documentul AVIZ (de expeditie).

17) Sa se afi§eze valoarea maxima, valoarea medie, valoarea minima §i valoarea totala pentru livrarile (IE= -1) de produse efectuate.

```
SQL> SELECT MAX (1.05*cant*pret) VZ MAX,
        AVG (1.05*cant*pret) VZ MED,
        MIN(1.05*cant*pret) VZ MIN,
        SUM(1.05*cant*pret) VZ TOTAL FROM
        produse, proddoc WHERE produse.
        codp=proddoc. codp
                  AND IE= -1
        GROUP B Y produse. codp;
```

VZ MAX	VZ MED	VZ MIN	VZ TOTAL
1.103E+09	735000000	367500000	1.470E+09
210000000	210000000	210000000	210000000
1.418E+09	1.418E+09	1.418E+09	1.418E+09

18) Sa se calculeze §i afi§eze profiturile rezultate din vanzari (IE= -1) cu comision de 5%

```
SQL> SELECTp.codp,
        pretH.95        PROFIT
        FROMproduse p, proddoc pd
        WHERE p. codp=pd codp AND
        IE= -1;
```

CODP	PROFIT
PI	3325000
P2	950000
PI	3325000
P4	2565000

19) Sa se afi§eze tranzactiile cu valoare mai mica decat cea mai mare valoare a unei tranzactii cu furnizorul 4

```
SQL> SELECT codt, valoare
        FROM documente
        WHERE valoare < ANY

                SELECT valoare
                FROM documente d, tranzactii t
                WHERE d codt=t. codt
                        ANDcodf='4'
```

CODT	VALOARE
T2	1.255E+09
T3	550000000
T3	550000000
T5	570000000
T5	570000000

20) Afi§ati produsele care au cantitatea mai mare decat cea mai mica cantitate a produsului "P4" *(min(cant)P4=200).*

```
SOL   SELECT codp, cant
      FROM proddoc
      WHERE
            Codp!= (P4'AND   cant > SOME
                              (SELECT DISTINCT cant
                              FROMproddoc
                               WHERE codp='P4')
      ORDER B Y cant DESC;
```

CODP	CANT
P2	500
PI	500
PI	500
P2	500
PI	300
PI	300

Cea mai mica cantitate a produsului 'P4' este de 200 bucati, astfel ca, cererea principals Tntoarce toate produsele, cu exceptia lui 'P4' (specificata explicit) care sunt Tntr-o cantitatea mai mare decat minimul cantitatii produsului 'P4' specificat.

Astfel, conditia '> ANY' Tnseamna *"mai mare ca minim"* iar '=ANY' este echivalent cu operatorul *IN.*

Cand se foloseste SOME/ANY, DISTINCT este frecvent utilizat pentru a Tmpiedica sa se selecteze liniile de mai multe ori.

21) Sa se afi§eze produsele care au cantitatea mai mare sau egala cu cea mai mare cantitate a produsului "P4" *(max(cant)P4=5200),* inclusiv produsul 'P4'.

```
SQl>  SELECT codp, cant
      FROMproddoc WHERE
      cant > = SOME
                  (select MAX (cant)
                  FROMproddoc
                   WHERE codp='P4')
      ORDER B Y cant DESC;
```

CODP	CANT
P2	500
PI	500
PI	500
P2	500
P4	500

Revista Virtuala
Info
MateTehnic
ISSN
2069-7988

ISSN-L
2069-7988
P4 P4 500
500

22) Sa se afis.eze, pentru fiecare document in parte, ce procent reprezinta produsele din totalul de produse de pe document.

*SQL> SELECTa.codd, a.codp, a.PROC/b.TOTAL*100 PROCENT*
FROM
(SELECT codd, codp, cant PROC
FROMproddoc GROUP BY codd, codp,cant) a,
(SELECT codd, SUM (cant) TOTAL
FROMproddoc GROUP BY codd) b WHERE a.
codd=b. codd;

CODD	CODP	PROCEN
10123	PI	50
10123	P2	50
10124	P3	14.285714
10124	P4	71.428571
10124	P5	14.285714
10125	PI	33.333333
10125	P2	66.666667
10126	PI	37.5
10126	P4	62.5
10127	P3	33.333333
10127	P4	66.666667
20123	PI	50
20123	P2	50
20124	P3	15.384615
20124	P4	69.230769
20124	P5	15.384615
20125	P3	33.333333
20125	P4	66.666667
30122	PI	33.333333
30122	P2	66.666667
30123	PI	37.5
30123	P4	62.5

23) Sa se afi§eze documentele avand valorile totale cuprinse Tntre 1.500.000.000 si 6.500.000.000 sau cele ca suntNIR-uri si Facturi.

*SQL> SELECT *from documente*
WHERE valoare BETWEEN 1500000000 AND 6500000000 OR
(dend=NIR'AND dend='FACT') ORDER BY data ASC;

CODD	DEND	DATA	CODT	VALOARE
20123	NIR	08-JAN-03	Tl	2.250E+09
10123	FACT	08-JAN-03	Tl	2.250E+09
30123	AVIZ	ll-FEB-03	T4	2.400E+09
10126	FACT	ll-FEB-03	T4	2.400E+09

24) Sa se afi§eze perioada de timp, in saptamani ramase pana la expirarea fiecarui produs. Saptamanile (cu perioadele interimare rezultate) se vor rotunji (prin functiile ROUNDsau TRUNC) la valorile Tntregi.

```
SQL> SELECT termen,
       ROUND ((termen-sysdate)/7) SAPT GARANTIE
       FROM produse;
```

TERMEN	SAPT GARANTIE
01-AUG-06	126
01-AUG-05	74
01-JUN-04	13
01-DEC-04	39
01-JUN-04	13

25) Sa se afi§eze denumirea §i valoarea documentelor Tmpreuna cu data incheierii lor, doar pentru tranzactiile Tncheiate in luna februarie 2005.

```
SQL> SELECTdend, valoare, data DATA INCHEIERII
       FROM documente WHERE TOJCHAR (data,
       'MM/YT) = '02/05';
```

DEND	VALOARE	DATA INCHEIERII
FACT	570000000	27-FEB-05
NIR	570000000	29-FEB-05

26) Sa se creeze un index nou pe atributul denumire produs (Denp) din tabela Produse.

```
SQL> CREATE INDEX prod idx ON
       produse (denp);
Index created.
```

27) Sa se afi§eze indecsji creati pentru tabela Produse §i daca asigura unicitatea.

```
SQl>  SELECT
      ICindexname, ICcolumnname,
      IC.column_position COLPOZ,
      IX. uniqueness
      FROM userindexes IX, userindcolumns IC
       WHERE IC. index_name=IX. indexname
      AND IC.table_name= 'PRODUSE ';
```

No rows selected.

28) Sa se §tearga indexul creat anterior.

```
SQl>  DROP INDEX prod idx;
```

Index dropped.

29) Sa se afi§eze restrictiile definite pentru tabela Tranzactii.

```
SQL> SELECT CONSTRAINT TYPE     TIP RESTR,
      CONSTRAINT NAME           NUME RESTR,
      STATUS                    STARE A RESTR
      FROM USER CONSTRAINTS WHERE
      TABLE NAME= 'TRANZACTII';
```

TIP RESTR	NUME RESTR	STAREA RES
C	NN DENT	ENABLED
C	CK DENT	ENABLED
P	PK CODT	ENABLED
R	FK CODF	ENABLED
R	FK CODC	ENABLED

30) Sa se afi§eze numele tabelelor create in schema proprie de obiecte

```
SQL> SELECT tablename FROM USER TABLES;
```

TABLE NAME

BONUS
CLIENTI
DEPT
DOCUMENTE
EMP
FURNIZORI
PRODDOC
PRODUSE

SALGRADE
TRANZACTII 10
rows selected.

31) Sa se afi§eze tipurile de obiecte create in schema proprie de obiecte

SQL> SELECT DISTINCT OBJECT TYPE
FROM USER OBJECTS;

OBJECT TYPE

INDEX
SEQUENCE
TABLE

32) Sa se redenumeasca tabela Clienti in tabela "ClientiRedenumiti".

SQL> ALTER TABLE clienti RENAME TO clientiredenumiti;
Table altered.
*SQL> SELECT * FROM ClientiRedenumiti;*

CODC	DENC	ADR	LOC	CONT	BANCA
1	CONN	PIPERA135	BUCURESTI	A1234567890	BRD
5	Flanx	Dorobantilor 130	Cluj	C1231231234	BCR

33) Sa se §tearga tabela ClientiRedenumiti §i sa se elibereze spatiul ocupat de aceasta.
SQL> TRUNCATE TABLE clientiredenumiti;

34) Sa se creeze un sinonim public pentru tabela Produse din schema de obiecte Student.

SQL> CREA TE PUBLIC SYNONYM prod FOR studentproduse;

35) Sa se creeze utilizatorul AGENT001 cu parola Agent.

SQL> CREATE USER AGENT001
IDENTIFIED BY agent;

36) Sa se creeze o serie de drepturi la nivel de sistem pentru utilizatorul AGENT001.

SQL> GRANT CREATE TABLE,
CREA TE SEQUENCE,

CREATE VIEW
TO AGENT001;

37) Sa se modifice parola utilizatorului AGENT001 cu "noua_parola"

SQL> ALTER USERAGENT001
IDENTIFIED BYnoua_parola;

38) Sa se creeze rolul AgVanz cu drepturile RESOURCE si CONNECT la nivel de si stem.

SQL> CREATE ROLE agvanz;
SQL> SET ROLE AgvVanz;

SQL> GRANT RESOURCE, CONNECT TO AgVanz;

39) Sa se ata§eze rolul AgVanz utilizatorului AGENT001.

SQL> GRANT AgVanz TO AGENT001;

40) Sa se anuleze drepturile primite de utilizatorul AGENT001 pe tabela Documente.

SQL> REVOKE ALL ON documente FROM AGENT001;

41) Sa se creeze tabela partitionata Vanzari (codt, data, suma) cu partitii pentru vanzarile din ultimele 3 luni.

SQL> CREATE TABLE
vanzari (codt varchar2 (5), data date, suma number (11))
STORAGE (INITIAL 100KNEXT50K) LOGGING
PAR TITION B Y RANGE (data)
(PARTITIONLUNA03 VALUES LESS THAN (4)
TABLESPACE TO,
PARTITIONLUNA02 VALUES LESS THAN (3)
TABLESPACE Tl,
PARTITIONLUNA01 VALUES LESS THAN (2)
TABLESPACE T2);

42) Sa se adauge noi tupluri din tabela Tranzactii in partitia LUNA02 din tabela Vanzari.

```
SQL> INSERT INTO vanzari
     PARTITION (LUNA02)
     SELECT T.codt,
     TOJCHAR (dataora,'MM-DD-YYYY% valoare FROM
     tranzactii T, documente D WHERE Tcodt=D.codt
     AND MONTHS_BETWEEN (data, sysdate)=2;
```

43) Sa se creeze un raport care sa afi§eze informatiile despre tranzactiile Tncheiate in ultimul an, documentele §i produsele aferente §i un total general.

```
SQL> SETPAGESIZE 200
SQL> SET LINESIZE100
SQL> SET FEEDBACK OFF
SQL> SET ECHO OFF SQL>
SET VERIFY OFF
SQL> COLUMN CODT FORMAT a5 HEADING 'Nr' SQL> COL
UMN DENT FORMA T al HEADING 'Tip' SQL>
COLUMNDATAORA FORMATalOHEADING 'Data' SQL> COL
UMN DEND FORMA T a4 HEADING 'Doc' SQL> COL UMN
CODD FORMA T 99999 HEADING 'Nr Doc' SQL>
COLUMNCODFFORMATa5HEADING 'Fz' SQL> COL UMN
CODC FORMA T a5 HEADING 'CV SQL> COL UMN VALOARE
FORMA T 99999999999
     HEADING 'Valoare Tranzactie' SQL> COL UMN CODP
FORMA T a5 HEADING 'Produs'
     COL UMN CANT FORMA T 99999 HEADING 'Cantitate'
SQL> SELECT T.codt, dent, dataora, dend,
     D.codd, codf, code, valoare, codp, cant
     FROM tranzactii T, documente D, proddoc P
      WHERE      Tcodt=Dcodt
                 AND Dcodd=P.codd
     ORDER BY T.codt, Dcodd, codp
```

N	T	Data	Doc	Nr Doc	Fz	Cl	Valoare	Produs	Cantitat
Tl	R	08-JAN-04	FACT	10123	3	1	2250000000	PI	500
Tl	R	08-JAN-04	FACT	10123	3	1	2250000000	P2	500
Tl	R	08-JAN-04	NIR	20123	3	1	2250000000	PI	500
Tl	R	08-JAN-04	NIR	20123	3	1	2250000000	P2	500
T	R	10-NOV-04	FACT	10124	4	1	1390000000	P3	100
T	R	10-NOV-04	FACT	10124	4	1	1390000000	P4	500
T	R	10-NOV-04	FACT	10124	4	1	1390000000	P5	100

CAPITOLUL 8
Sisteme de gestiune a bazelor de date distribuite (SGBDD)

8.1.Caracterizare generala

Definiţie

Un sistem de gestiune a bazelor de date distribuite (SGBDD) este o colecţie de programe complexe şi procese peste o reţea de echipamente, în care fiecare nod are o autonomie locală (propriul ceas şi memorie interna), iar nodurile, în ansamblul lor, sunt transparente pentru utilizator în ideea realizării obiectivului urmărit de sistemul distrubuit, care cooperează în rezolvarea anumitor probleme.
SGBDD permite accesarea simultană şi modificarea datelor de pe mai multe calculatoare diferite, folosind reţeaua de calculatoare.
In esenţă un SGBDD este un sistem în cadrul căruia mai multe baze de date locale sunt legate printr-o reţea de comunicaţie, astfel încît, datele de pe orice calculator pot fi accesate de orice utilizator din reţea.

Rol –Bazele de date, care alcătuiesc colecţia, numită *bază de date distribuită* se găsesc în nodurile reţelei de calculatoare, fiecare bază de date este administrată şi întreţinută separat, de propriul său SGBD, ca şi cum nu ar fi legată în reţea.
Rolul SGBDD pentru dezvoltarea aplicaţiilor. Cîteva aspecte privind SGBDD.

- multe companii îşi desfaşoară activitatea prin sedii, filiale, distribuite geografic în mai multe zone dintr-o ţară sau din mai multe ţări.
- fiecare sediu îşi are propria organizare şi activitate care se desfaşoara local.
- la o anumită perioada, sediile trebuie să comunice fie intre ele, fie cu sediul central pentru schimb de informaţii.
- rezolvarea aplicaţiilor pentru companiile care au o astfel de organizare se foloseşte bazele de date distribuite.
- bazele de date locale (noduri) are un mare grad de independenţă care asigură un nivel de protecţie asupra avariilor ce pot apare la calculatoarele din reţea.
- administratrea locală se realizează mai uşor, deci se păstrează controlul şi responsibilitaţile administrative.
- limitarea numărului de tranzacţii distribuite de pe un nod sau stabilirea de blocare în cazul accesului concurrent.

Componente

SGBDD are o serie de componente software, anume:

1. componentele de comunicaţie;
2. SGBD-urile locale;
3. SGBD-ul distribuite.

SGBD –urile locale sunt cele care descriu şi manipuleaza datele din bazele de date, care există pe staţiile de lucru din nodurile reţelei de calculatoare.
Ele pot fi de tipuri diferite sau de acelaşi tip (de exemplu: Oracle …)
SGBD distribuit (SGBDD), este un sistem software complex care asigură gestionarea bazei de date distribuită în mai multe noduri ale reţelei de calculatoare.
El realizează accesarea datelor în urma mai multor cereri de regăsire.

- crează şi ţine la zi un dicţionar de date global, care conţine informaţii despre BDD.

- informaţiile pot fi consultate de orice utilizator şi care se referă la structura, distribuirea (localizarea), accesbilitatea şi modul de utilizare a datelor din BDD.

8.2 Obiective

1. asigurarea unei redundante minime şi controlate a datelor.

Distribuirea în reţea poate genera o creştere a redundanţei, în sensul memorării unei copii a unei baze de date sau a unei părti din ea în două sau mai multe noduri (replicarea segmentarea datelor).

2. asigurarea unor facilităţi de utilizare.

Intr-o bază de date distribuţia (BDD), sistemul de gestiune a bazelor de date distribuite asigură transparenţa localizarii informaţiilor pe calculatoarele din reţea. Acest lucru în principu utilizatorul nu ştie şi nu trebuie să stie, pe ce noduri se găsesc datele de care are nevoie.

3. asigurarea securitaţii datelor.

Intr-o reţea de calculatoare, problema securităţii datelelor are aspecte suplimentare datorită distribuirii datelor şi existenţei unui numar mare de utilizatori care pot accesa datele.

Software-ul de reţea poate asigura un nivel de securitate, sistemul de oprare pe un alt nivel, iar SGBD-ul local un alt nivel, după care se adaugă un nivel de securitate global (la nivelul bazei de date distribuite) asigurat de SGBDD.

- autorizarea şi controlul accesului la date distribuite prin parole
- folosirea viziunilor permite definirea unor partiţii logice ale BD

4. asigurarea coerenţei şi integrităţii datelor.

SGBD-urile locale asigură acest obiectiv la nivelul fiecărui nod.

La nivelul întregii baze de date distribuite trebuie să asigure respectarea unor restricţii de integritate la operaţiile de actualizare.

In caz de incident se păstrează şi apoi se reface (salvare/restaurare) modificarile care au apărut după producerea incidentului.

5. asigurarea partajabilităţii datelor.

In situatia în care mai mulţi utilizatori folosesc simultan baza de date distribuită, apare o mare importanţă pentru SGBDD, localizarea datelor.

Fluxul activitaţilor este descris în continuare.

Un utilizator lansează o cerere de regăsire de la o staţie locală (nod în reţea). Cererea de regăsire este preluată de SGBDD care, cu ajutorul dicţionarului de date, localizează nodul unde sunt stocate datele necesare. Sunt trei situaţii posibile:

- cererea poate fi satisfacută în întregime de SGBD-ul local de unde ea a fost lansată (crere locală);
- cererea poate fi satisfacută în întregime de un SGBD local de la o altă staţie decăt cea de unde ea a fost lansată;
- Crerea este transferată de SGBDD nodului respectiv pentru prelucrare (cerere la distantă);
- cererea poate fi satisfacută numai preluînd date de pe mai multe staţii

6. asigurarea administrăii şi controlului datelor.

SGBDD asigură o viziune de ansamblu, asupra întregii baze de date distribuite. Pentru acest lucru el are o serie de componente care-l ajută să supervizeze şi să controleze baza de date şi informaţii despre modul de utilizare a datelor distribuite. Dacă SGBD-urile sunt eterogene, atunci sarcina SGBDD este mai usoară, în caz contratr este mai complexă.

Regulile lui Date

Un SGBDD pentru a fi un *sistem de gestiune a bazelor de date distribuit* ar trebuie să respecte cele 12 reguli intocmite *C.J.Date.*
Aceste reguli sintetizează principalele motive pentru care este necesară distribuirea datelor pe mai multe calculatoare într-o reţea.
In cele ce urmează prezentăm cele 12 reguli ale lui Date :
R1: *Atunomia locala* presupune ca fiecare nod pe care se gasesc date va avea controlul complet asupra acestora şi, va fi independent de celelalte noduri;
R2. *Nu exista server central* de care să depinda intregul sistem de baze de date distribuite (BDD) ;
R3. *Activitatea dintr-o baza de date distribuita* este continua, fără intreruperi pentru intreţineri sau alte operaţii;
R4. *Transparenţa localizarii şi independenta* de localizare presupune ca nici un utilizator sau program nu stie unde sunt amplsate datele de care are nevoie;
R5.*Independenta fragmentarii* presupune ca SGBDD va reconstitui automat o colectie de date care a fost fragmentata;
R6 *Independenta replicarii* presupune ca operatia de duplicare a datelor este o functie automata a SGBDD.
Utilizatorii şi programele nu trebuie sa cunoasca faptul ca datele au fost replicate şi nici că se lucreaza cu o anumita copie;
R7.*Interogarile distribuite* sunt prelucrate printr-un optimizator al SGBDD pentru a obţine rezultate cat mai bune;
R8.*Tranzactiile distribuite*,ca sarcina a SGBDD,presupun actualizari ale unei baz de date aflata pe mai multe noduri dintr-o retea;
R9.*Independenta* fată de hardware presupune faptul ca nodurile pe care se gasesc datele pot fi calculatoare de diferite tipuri si puteri eterogene;
R10.*Independenta fată de software* presupune, că nu trebuie să aibă importantă sistemele de operare care exista pe noduri ;
R11.I*ndependenţa de reţea* presupune faptul ca diferitele protocoale utilizate in retea nu trebuie sa afecteze functionarea bazei de date distribuite (BDD);
R12.*Independenţa faţă de SGBD* presupune,ca la nivelul de nod local pot rula diferite SGBD-uri.

8.3 Tipuri de baze de date distribuite

Bazele de date distribuite (BDD) sunt rezultatul aplicării prelucrării distribuite, evoluţiile în sistemele de prelucrare a datelor, sistemele în timp real, disponibilităţile mini, micro, au constituit apariţia sistemelor de prelucrare distributivă.
Definitie
O baza de date distribuita este o colectie de date integrate din punct de vedere logic, dar fizic distribuite pe staţiile unei reţele de calculatoare.
Fiecare staţie are o autonomie de prelucrare care îi permite să realizeze aplicaţii locale, iar pe de alta parte sa participe la executia aplicaţiilor globale care necesita accesarea datelor din mai multe staţii.

Din punct de vedere logic există o singură bază de date cu care utilizatorul interacţioneaza la fel ca şi în cazul bazelor de date centralizate, iar din punct de vedere fizic are loc o partiţionare a bazei de date pe staţiile reţelei de calculatoare.

Avantajele specifice BDD sunt :

- Posibilitatea extinderii bazei de date prin adaugarea de noi structuri fără a fi afectate aplicatiile existente şi cu impact minim asupra structurii predefinite ;
- Creştera numarului de aplicaţii realizate local ;
- Creşterea gradului de paralelism in executarea aplicatiilor ;
- Creşterea sigurantei sistemului (caderea unei statii nu afecteaza intregul sistem) ;
- Disponibilitatea permanenta a datelor prin existenta copiilor pe alte statii (replicarea datelor)

Clasificarea bazelor de date distribuite

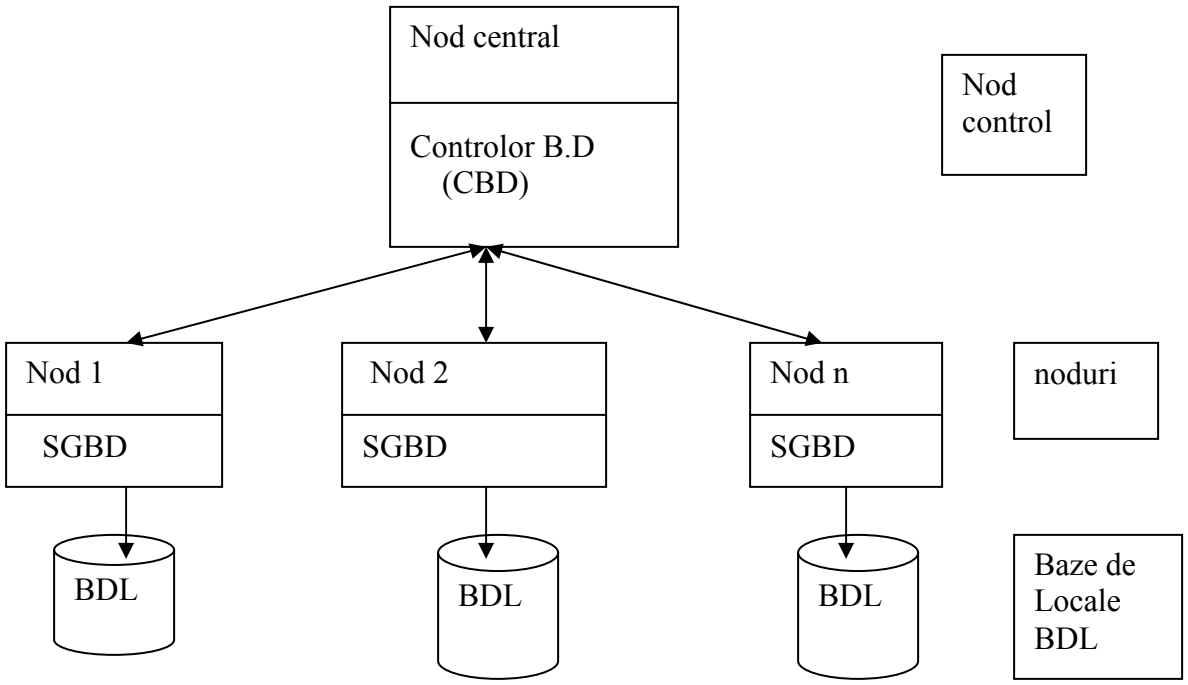
Exista trei tipuri de baze de date distribuite (BDD).

1. Baze de date distribuite totale (BDDT) , cînd toate datele se află în nodurile reţelei, existînd şi situatia cînd nodul central care are rolul de a controla funcţionarea bazei, nu conţine date

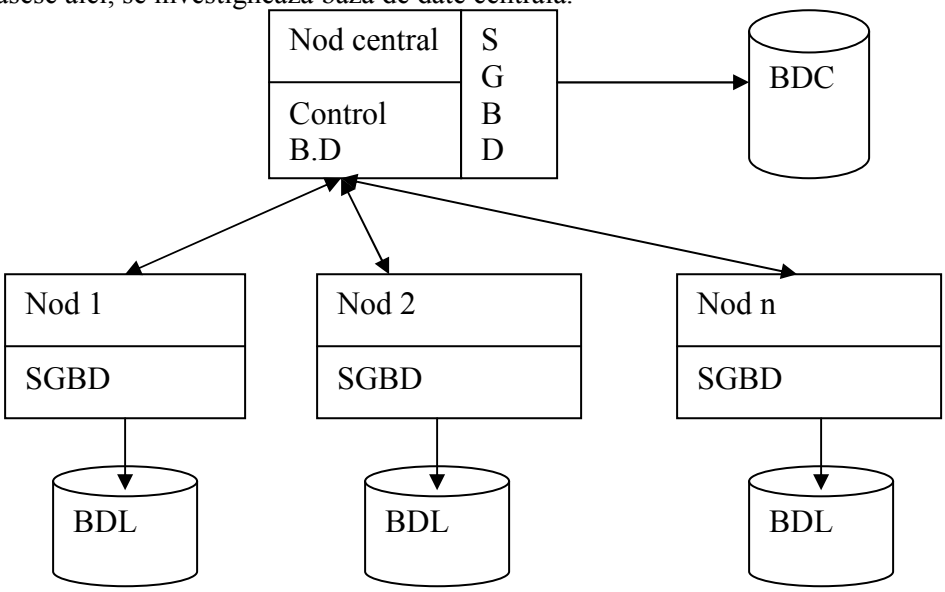
Baza de date este formată din suma bazelor de date locale (BDL), gestionate de un controler (CBD)

Accseul statiilor legate la un nod, la datele din baza de date locala (BDL), reface prin intermediul procesorului local, iar la datele aflate la distanta, prin controlerul bazei de date (CBD).

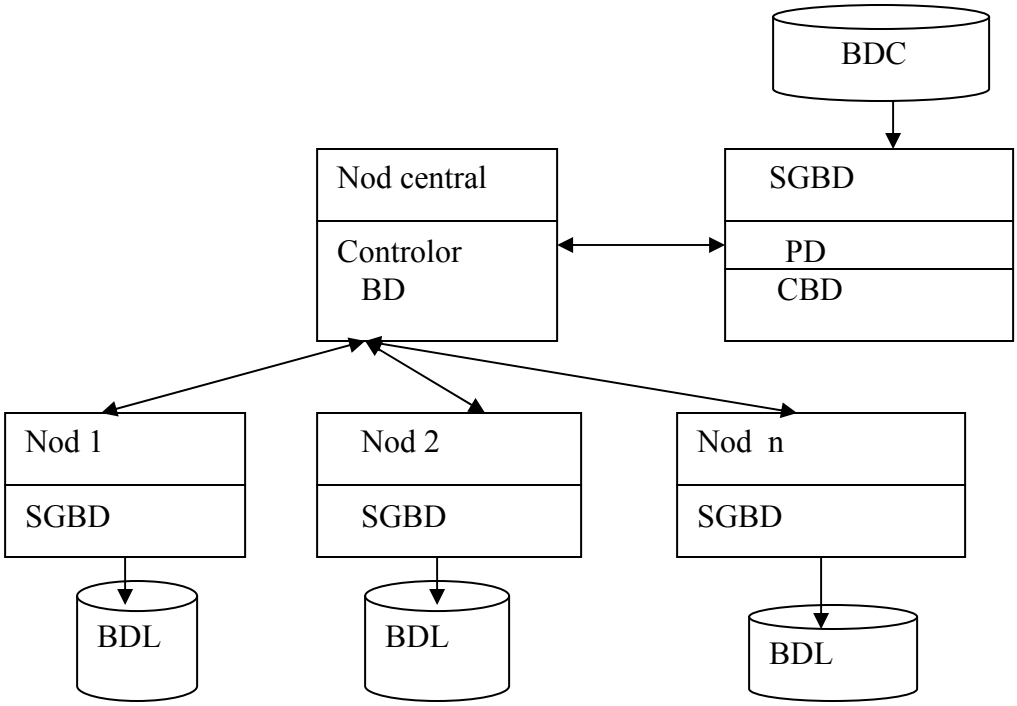
Figura baza de date totala distribuita



2. Baza de date distribuita central (BDDC) care conţine toate datele din sistem şi un numar de baze de date locale compuse din datele locale.
In acest caz baza de date este unica exista fizic in nodul central.
La accesarea datelor se cauta in baza de date locala (BDL) şi in cazul cand nu se gasesc aici, se investigheaza baza de date centrala.



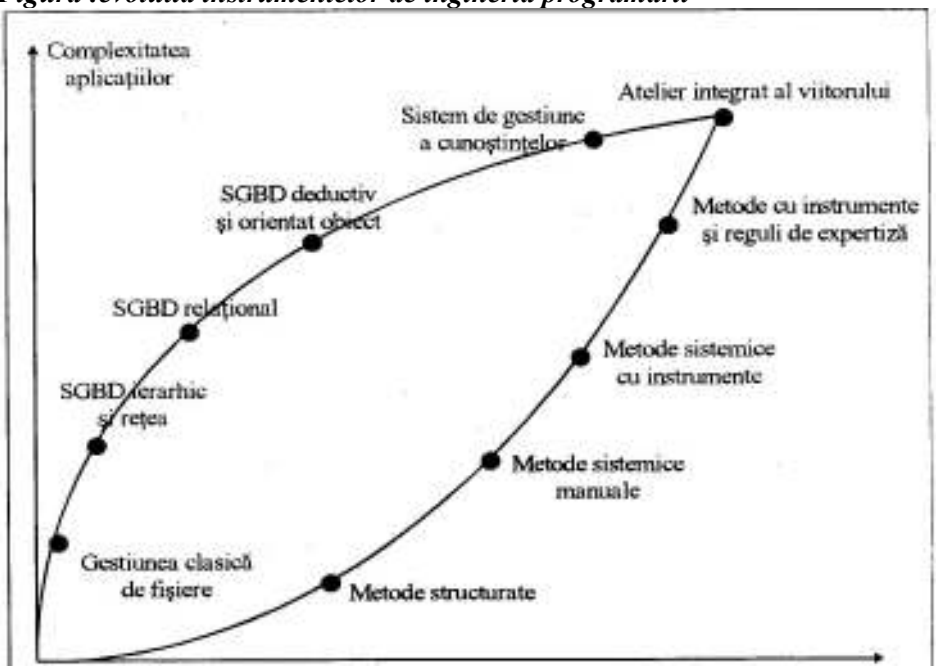
3. Baza de date paralel distribuita (BDPD), se asemana cu bazele de date central distribuite, dar accesul la date nu mai este gestionat direct de procesorul central şi prin intermediul unui procesor specializat (dedicat).
Accesul la un nod oarecare se poate face prin procesorul dedicat controlului bazei de date centrale.



Pentru alegerea unui tip de baza de date distribuita trebuie avuta in vedre urmatoarele aspecte :

- volumul informaţiilor de stocat ;
- frecventa accesărilor şi actualizărilor ;
- raportul dintre utilizarea globală şi cea locală a datelor;
- distribuirea optimă a bazelor de date în reţea;
- alocarea datelor;

Figura :evolutia instrumentelor de ingineria programarii



Pentru a se elimina varietatea de produse software care se folosesc intr-o reţea şi care concură la realizarea şi utilizarea unei baze de date distribuite, s-au conceput o serie de produse :**middleware »**
Ele sunt interfeţe care pot fi utilizate de toate programele de aplicaţii pentru a accesa datele distribuite.
Exemplu, este produsul « SQLMiddleware, care localizeaza datele, converteste cererile de regasire, le transmite in noduri si intoarce rezultatul utilizatorului., el permite acces la datele din noduri gestionate de DB2, Oracle.

8.4 Arhitectura client/server
Tehnologia client/server, ca practică în reţele de calculatoare, este strîns legată de tehnologia bazelor de date distribuite(BDD).
1.Caracterizare generala
Definitie
Arhitectura client/server este un ansamblu de patru componente principale :

- un server, un client şi o reţea care conecteaza calculatorul client la cel server pentru a colabora la indeplinirea sarcinilor
- delimitarea neta dintre serviciile de prezentare şi cele de manipulare a informaţiilor ;
- flexibilitate, punerea in funcţiune a unui mecanism de asigurare a securitaţii şi integritaţii pentru datele rezidente pe servere
- arhitectura deschisa, în sensul unei multitudine de platforme (mainframe, mini, micro) şi de produse (aplicatii-program).

Scopul arhitecturii client/server se interfereaza cu cel al bazelor de date distribuite, păstrându-se însă suficiente aspecte specifice pentru fiecare.

- dezvoltarea de aplicatii care necesita date situate pe calculatoare deferite, in diferite puncte geografice.

Aceste date pot fi in formate compatibile sau aceleasi dar si in formate incompatibile.
Eemplu: dezvolatarea aplicatiilor in diferite puncte ale unei tari.

- dezvoltarea unor aplicatii interactive, adaptate cerintelor utilizatorilor.

Acestia vor utiliza aplicatiile accesand datele de pe mainframe-uri la fel de usor ca datele de pe statiile de lucru

- cantitatea de date stocata,prelucrata şi regasită este foarte mare,de mare complexitate

In concluzie, scopul arhitecturii client/server este de-a permite dezvoltarea aplicatiilor complexe, de la calculatoare diferite situate la distanta.
De exemplu: Aplicatii client/server: visualfoxpro 9.0 /Oracle 9i2 ;
Java jakarta etc….
Tehnologia client/server oferă urmatoarele aspecte :

- Conectarea diferitelor tipuri de calculatoare (micro/mainframe) ;
- Colaborarea diferitelor categorii de utilizatori ;
- Tratarea unitara a datelor ;
- Asigurarea protectiei şi securitaţii datelor distribuite in reţea ;
- Utilizarea mainframe-urilor prin programme la fel de prietenoase ca cele de pe micro ;
- Mainframe-ul devine server

Activitatea client/server
In colaborarea dintre client si server ,în reţea exista trei situaţii *:client pasiv,server pasiv si client/server*

1. *client pasiv* este un terminal cu o putere de calcul limitata conectat la un server

2. s*erver pasiv* este un calculator mai puternic dintr-o retea locala (LAN)

Într-o retea locala aplicatiile se executa pe calculatoarele client, iar serverul este solictat doar pentru anumite operatii

În functie de aceste opratii,serverul poate fi :

- de fisiere, atunci cand pe discul său sunt stocate fişierele clienţilor
- de tipărire

Principalele arhitecturi ale aplicaţiilor in care sunt implicate bazele de date.

Una dintre cele mai importante proprietati ale bazelor de date tine de caracterul public, de accesibilitatea lor.

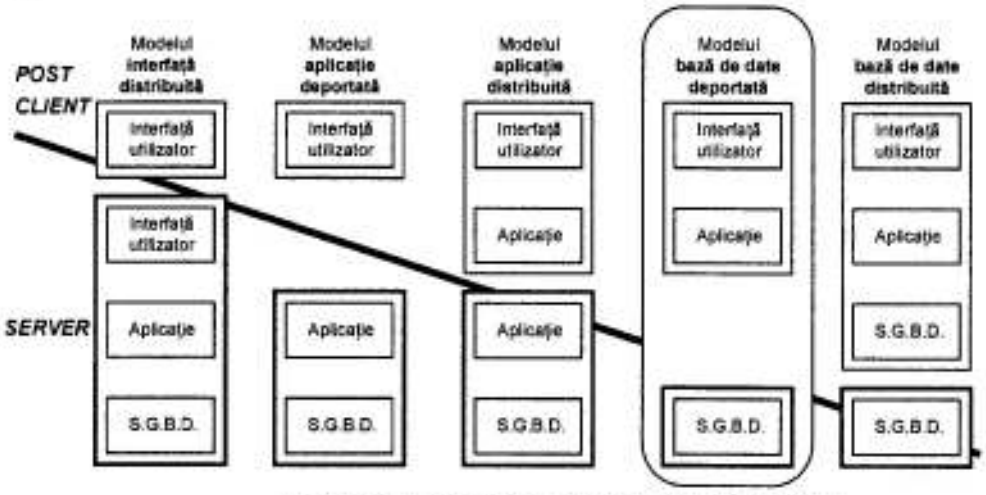
Orice sistem client/server este alcatuit din trei componente :

- Interfaţa cu utilizatorul (sistem de operare/mediu grafic)
- Aplicaţia(prelucrarile)
- Sistemul de gestiune a bazelor de date (SGBD).

In practica exista mai multe modalitatii de repatizare a funcţilor intre client şi server.

Figura *cinci moduri de repatizare a funcţiilor între client şi server.*

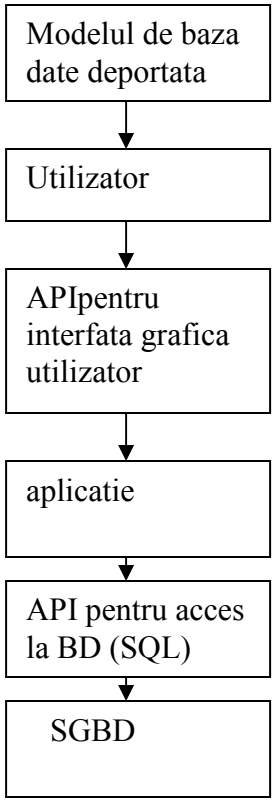
- Modelul interfata distribuita –este impartit intre platformele client si server, iar aplicatia SGBD-ul, ambele, rezidente pe server
- Modelul aplicatie deportata-este plasat pe platforma client, iar aplicatia SGBD sunt situate pe server
- Modelul aplicatie distribuita-presupune localizarea interfetei pe calculatoarele client, a SGBD-ului pe server
- Modelul de date deportata
- Modelul baza de date distribuita

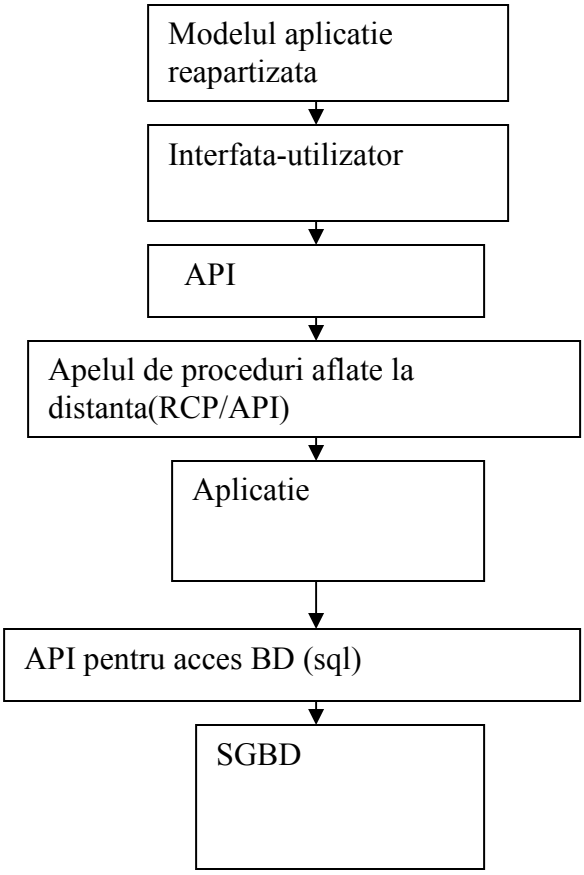


Cinci moduri de repartizare a funcţiilor între client şi server (*sursa:* Gartner Group)

Rolul middleware-ului in sistemele client-server

Dialogul propriu-zis dintre client şi server cade sub incidenţa unei componente, construit pe tipologia c/s:**middleware-ul.**





În noile condiţii tehnologice,sistemele client/server inglobeaza atăt aplicaţiile ce au la baza clasica arhitectura bazata pe interfata grafică utilizator(GUI),căt şi aplicaţiile web-centrice.

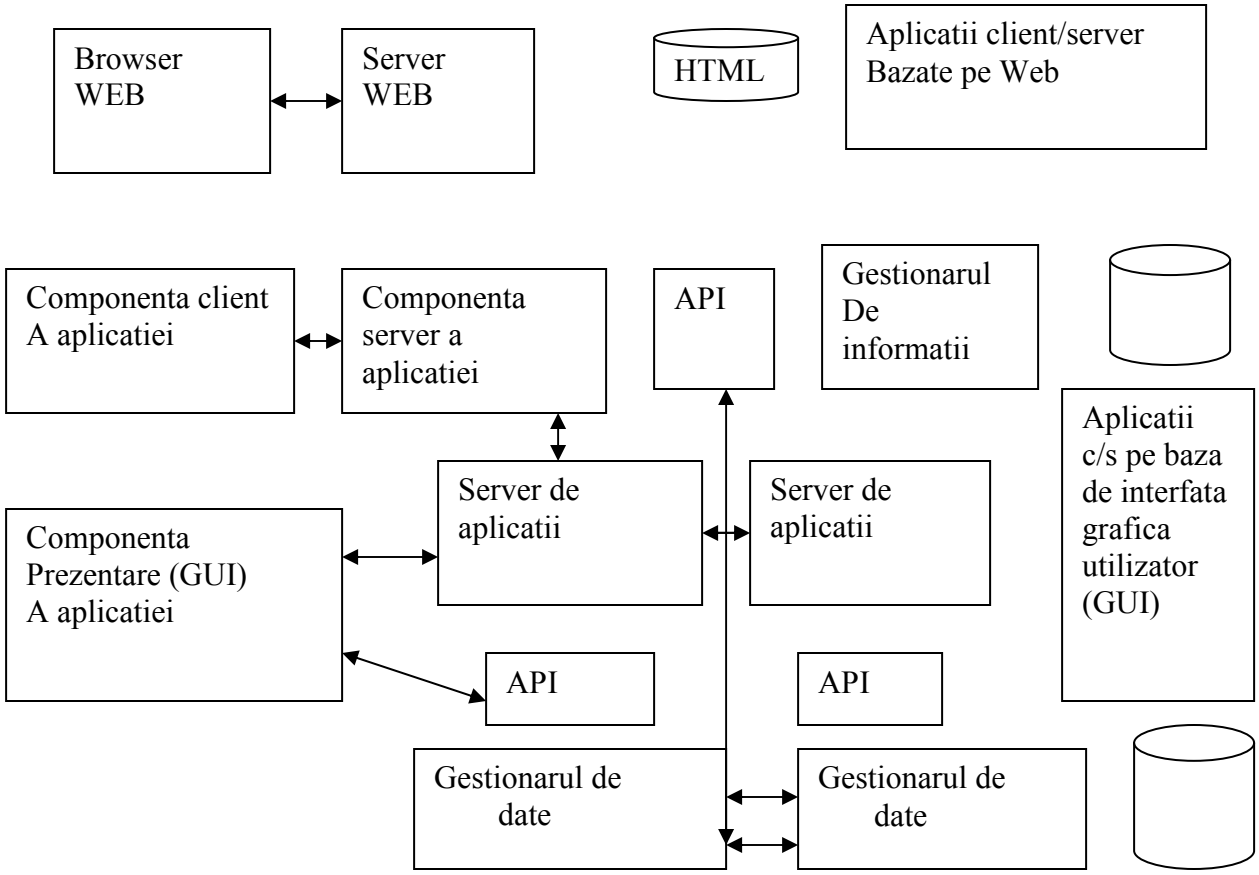


Figura cadrul general al unui sistem client-server actual

8.5. Clasificarea arhitecturilor unui SGBDD

O arhitectura distribuita presupune existenta unor baze de date multiple, care se gasesc pe calculatoare distincte şi a unor aplicaţii care manipuleaza datele de la diferite staţii de lucru locale, cu ajutorul unor SGBD-uri.

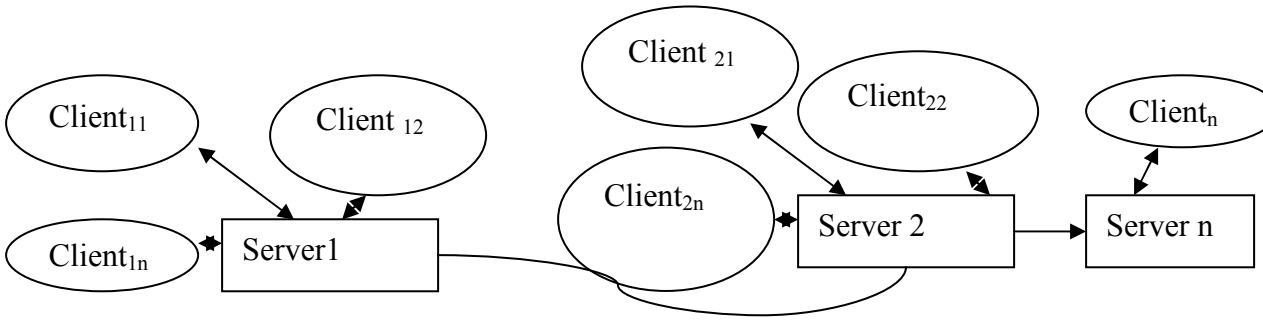


Figura :Arhitectura distribuita

SGBDD utilizeaza o serie de notiuni care apar intr-o arhitectura distribuita :

Reţeaua de calculatoare : este un ansamblu de componente hradware si software care asigura o legatura intre nodurile retelei
Nodul este un sistem de calcul dintr-o retea care indeplineste rolul de client, server sau ambele.
Serverul este un produs software care se instaleaza pe un calculator :Windows server 2003, Oracle 9i2…,SQL server 2005
Definiţie
Fiecare nod (baza de date) al unei baze de date distribuite este administrat separat si independent de celelalte, ca şi cum bazele de date componente nu ar fi legate în reţea.
Iata cateva dintre raţiunile utilizării bazelor de date distribuite :

- Nodurile unui sistem distribuit pot reproduce modul de organizare a companiei.
- Prin pastrarea la nivel local a controlului si a responsabilitatilor administrative, domeniul local devine mai usor de administrat. Administrarea centralizata a unei arhitecturi distribuite la nivel planetar ar fi o sarcina dificila.
- Independenţa bazelor de date locale asigura un nivel de protectie impotriva defectarii calculatoarelor din celelalte noduri.
- Actualizare distribuita-actualizeaza datele situate in doua sau mai multe noduri.
- Reţea- reprezinta totalitatea echipamentelor hradware şi software care leaga intre ele toate componentele bazei de date distribuite (BDD).
- Obiect al schemei-care este un tabel este accesibil din toate nodurile care formeaza o BDD
- Server-Un server este un produs soft care gestioneaza o baza de date.

Arhitectura distribuita a bazei de date Oracle sugereaza integrarea tuturor bazelor de date care sunt membre ale arhitecturii distribuite sub forma unei singure baze de date logice.
Administratorul bazei de date este responsabil de proiectarea bazei de date, efectuarea copiilor de siguranţa şi reconstituirea.
Comunicarea prin reţea
Administratorul bazei de date si cu administratorul reţelei trebuie sa asigure conectarea in retea a tutoror calculatoarelor, respectiv instalarea programului SQL*NET.
Nume globale
In arhitectur bazei de date distribuite, în fiecare dintre aplicaţiile distribuite, site-urile distribuite, toate numele obiectelor trebuie sa fie global unice.
Numele global al unei baze de date este compus din doua parti :componenta nume si componenta domeniu de retea.
Atunci can se creaza o baza de date distribuita,trebuie ales un nume care sa fie unic in toata reţeaua de baze de date si,respectate urmatoarele :

- componenta nume a unei baze de date globale reflecta continutul BD
- nu trebuie să se precizeze o localizare, numele domeniului reţelei trebuie sa respecte convenţiile internet.

Transparenţa
Sistemul de baze de date distribuite trebuie conceput în aşa fel încăt locatiile fizice ale unei baze de date distribuite sa fie transparente utilizatorilorilor finali.
Transparenţa locatiilor asigura urmatoarele avantaje :

- Se simplifica accesul la datele situate la distanta, deoarece nu este necesar ca utilizatorii să cunoasca localizarea obiectelor.

- Obiectele pot fi mutate fara ca acest lucru sa aiba vreun impact asupra programatorilor si utilizatorilor.

Transparenta locatiei nu se realizeaza automat,ea poate fi asigurata prin folosirea mai multor metode ,printre care :

- Se creaza o vedere definita local,un sinonim loal poate ascunde o legatura la o baza de date

Clientul este o aplicatie dintr-o reţea de calculatoare, de exemplu: *Visual foxpro 9.0,* aplicatii de tip client.
Interogare de la distanta este regasirea de date dintr-una sau mai multe tabele, ale unei baze de date, care se gasesc in acelasi nod situat la distanta fata de nodul unde s-a emis cererea de regasire.
Tranzactia la distanta este o secventa de una sau mai multe instructuini care face referire la acelaşi nod situat la distanta fata de nodul a fost lansata tranzactia.
Actualizare la distanta este o operatie de tinere la zi a datelor dintr-una sau mai multe tabele situate in acelasi nod la distanta fata de nodul de unde a fost initiata actualizarea.
Interogarea distribuita este regaasirea de date din doua sau mai multe noduri distincte
Tranzactia distribuita este o secventa de una sau mai multe instructiuni care face referire la datele din doua sau mai multe noduri distincte.
Exemplu de SGBDD care gestioneaza o arhitectura de baze de date distribuita este sistemuil Oracle9i2….
Aceasta integreaza toate bazele de date membre ale unei arhitecturi distribuite sub forma unei singure baze de date logice distribuita.

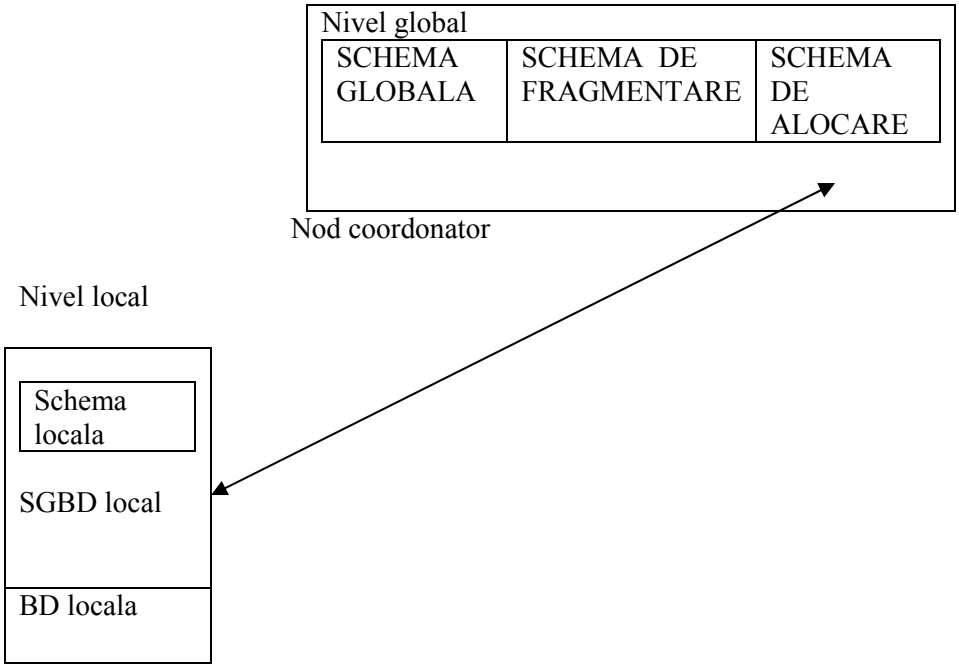
8.6 Clasificarea arhitecturilor :ANSI/SPARC ,IBM

Diferitele SGBDD care sunt în explotare curentă nu respecta o arhitectura unică. Pentru aceasta sunt implementate variante ale unei arhitecturi de referintă, astfel de arhitecturi pentru SGBDD,au fost elaborate de ANSI/SPARC şi IBM.

8.6.1. Arhitectura ANSI

ANSI este un organism american de standardizare care a elaborat, una din arhitecturile de referinţa pentru SGBD-uri.
In cazul sistemelor distribuite, arhitectura ANSI este concepută pe doua nivele :*un nivel global şi un nivel local.*

l Figura :arhitectura SGBDD de tip ANSI.



Nod cooperant
În această arhitectura de SGBDD exista un singur nod (statie) la nivelul global si mai multe noduri (statii) la nivelul local.
SGBDD trateaza o schema globala,referitoare la toate datele distribuite in retea,şi are in vedere mai multe scheme locale, referitoare la datele de pe fiecare staţie.

- Nivelul global al unui SGBDD are rolul de a asigura o tratare de ansamlu a bazei de date distribuite,in aceasta situate sunt integrate toate datele din bazele de date locale.

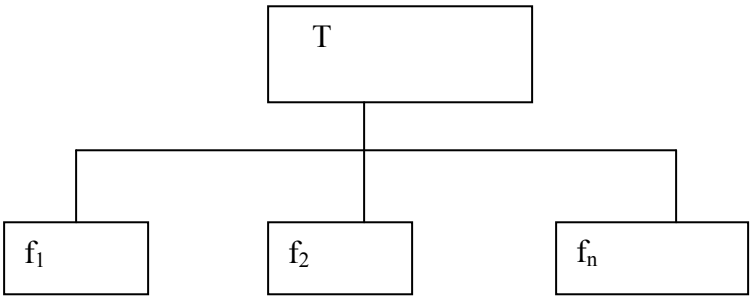
Integrarea se realizeaza cu ajutorul celor trei tipuri de scheme care sunt implementate de SGBDD la nivelul global: schema globala, schema de fragmentare şi de alocare.

Schema globală defineste şi descrie toate informaţiile din baza de date distribuita in reţea
Pentru descriere se foloseste unul din modelele de date fundamentale utilizate in bazele de date *:arborescent, retea, relational, orientat pe obiecte.*
In sens general, vom spune ca schema globala descrie un set de colectii globale şi legaturile dintre ele.
Daca de exemplu considerăm că SGBDD implementează modelul relaţional atunci schema globala descrie un set de tabele numite globale şi legaturile dintre ele.
Fiecare dintre tabelele globale se impart logic în fragmente.
Fragmentele sunt parţi disjuncte ale unei tabele globale şi un fragment nu poate interveni din mai multe tabele.

Schema de fragmentare descrie legaturile dintre o colectie globala si fragmentele sale.

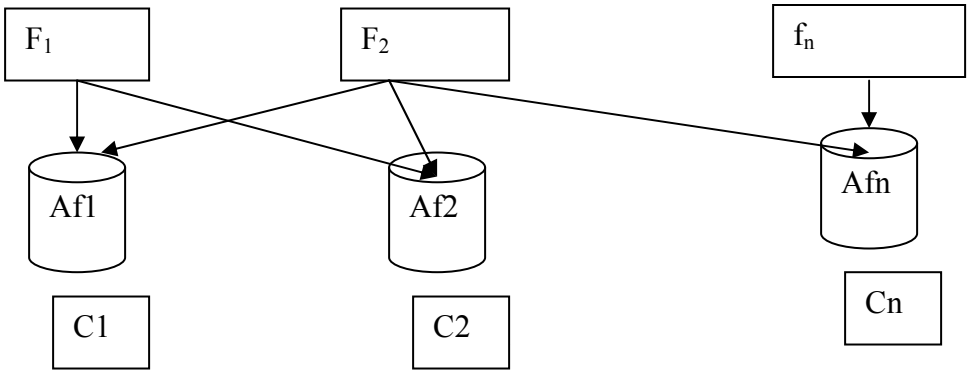
Figura: Schema de fragmentare

Tabela globala



fragmente

Schema de alocare descrie modul de distribuire a segmentelor pe calculatioarele (noduri) din retea.
În acest mod fiecare segment va avea o alocare fizica pe unul sau mai multe calculatoare, în aceasta schemă se introduce o redundanţa controlată.
Pe de alta parte, pe un calculator pot exista segmente din mai multe tabele globale,în aceste conditii, schema de alocare este de tipul multi la multi şi avand forma de retea din figura schema de alocare.



Unde :f_i =fragmentele tabelei globale T
AF_i=alocarea fizica,C_i=calculatoare din retea.
Exemplu de schema de alocare.

In acest exemplu, fragmentul f1 va fi alocat fizic atât pe calculatorul c1, c2, fragmentul f2 pe calculatoarele c1, c2……cm.
Totalitatea fragmentelor dintr-o anumita tabela globala care sunt alocate pe acelaşi calculator formeaza imaginea fizica a tabelei pe calculatorul respectiv.
De exemplu, imaginea fizica a tabele T pe calculatorul C1 este data de fragmentele f1, f2, iar pe calculatorul c2 este aceeaşi imagine.
Localizarea optima a datelor in nodurile unei reţele de calculatoare revine proiectantului şi administratorului bazei de date.

Nivelul local al unui SGBDD reuneşte toate bazele de date locale distribuite pe calculatoarele din retea.
Fiecare dintre aceste baze de date este trata ca centralizata cu ajutorul a trei componente :*schema locala,SGBD-ul local,baza de date locala.*
Schema locala are rolul de a realiza legatura intre imaginea fizica si datele locale, aceasta schema depinde de tipul SGBD-ului local.
SGBD-ul local implementeaza schema locala si are rolul de a descrie si gestiona datele de pe calculatorul local.
La nivel local, toate SGBD-urile utilizate pot fi de acelasi tip în cazul omogen, sau de tipuri diferite.
Baza de data locala contine datele locale conform imaginilor fizice, şi este organizata după un alt model de date acceptat de SGBD-ul local.

8.6.2. Arhitectura IBM

Firma IBM,are un puternic colectiv pe domeniul bazelor de date distribuite, care a elaborat o arhitectură de referinţă pentru BDD, numita *DRDA (Distributed Relational Database Arhitecture).*
Arhitectura are la bază deschiderea bazelor de date distribuite spre dezvoltari succesive, în acest scop sunt prevazute patru niveluri posibile de dezvoltare progresiva.
Cereri la distanta, tranzactii la distanta, tranzactii distribuite, cereri distribuite.

1.*cererile la distanta* semnifica faptul ca cerere este o tranzacţie separata, care se indreapta spre un singur nod, care este adresata pentru o BD de pe un nod aflat la distanta

2.tranzacţiile la distanta semnifica faptul ca fiecare tranzacţie poate accesa un singur nod de mai multe ori

3.tranzacţiile distribuite semnifica faptul ca intr-o singura tranzactie se permite accesul la mai multe baze de date situate in noduri diferite

4.cererile distribuite semnifica faptul ca printr-o singura cerere se pot accesa datele aflate pe mai multe noduri din retea.

Transparenta distributiei
Unul dintre cele mai importante obiective ale unui SGBDD este asigurarea distribuirii datelor, ca facilitate de utilizare.
Transparenta la nivel de fragmentare este asigurata de SGBDD în sensul ca utilizatorii lucreaza cu colectiile globale.
Transparenta la nivel de alocare este asigurata de SGBDD în sensul ca utilizatorii lucreaza cu fragmentele.
Transparenta la nivel local este asigurata de SGBDD în sensul ca utilizatorii lucreaza cu un SGBD local.

8.7.Realizarea distribuirii datelor

Sarcina SGBDD este de a asigura distribuirea datelor, atăt a celor de baza căt şi a celor auxiliare
In acest sens, prezentam cum realizeaza un SGBDD distribuirea informaţiilor din baza de date propriu-zisa şi respectiv din catalogul bazei de date.

8.7.1 Distribuirea datelor din baza de date

a) Distribuirea prin fragmentare

Fragmentarea este operaţia logica de descompunere a colecţiilor globale, dintr-o baza de date distribuita, în părţi disjuncte numite fragmente .SGBDD realizeaza fragmentarea prin intermediul unor operatori speciali aplicati colecţiilor globale pentru a realiza fragmentarea, SGBDD trebuie sa respecte cel putin trei reguli: *completitudinea, reconstructia şi disjuncţia.*

- completitudinea- este condiţia ca intreaga colecţie globala să fie descompusă in fragmente
- reconstrucţia este regula care cere ca orice colecţie globala să poata fi recompusă din fragmentele ei
- disjuncţia este condiţia ca fragmentele in care se descpune o colectie sa fie exclusive.

b)Distribuirea prin replicare

Replicarea este operatia de stocare a unor portiuni dintr-o baza de date, sub forma de copii, pe mai multe calculatoare dintr-o retea.

8.7.2. Replicarea bazelor de date (RBD)

În continuare se prezintă conceptul de replicare a bazelor de date şi rolul pe care acesta îl deţine în cadrul activitaţilor desfasurate.

Realizarea acestora activitaţii presupune o serie de factori, care sunt rezumati în eficienţa economica, care are drept rezultat extinderea fizica a societăţii economice, iar penetrarea altor pieţe, presupune infiinţarea de noi filiale (sedii).

Necesitatea transferului rapid de informatii a dus la aparaţia reţelelor de calculatoare.

De exemplu, se presupunem ca intr-o organizaţie de dimensiuni mari, un numar semnificativ de angajati trebuie să lucreze ce aceleaşi pachet de date.

Soluţia cea mai eficientă este încarcarea datelor pe un calculator central şi oferirea pe alte calculatoare, accesarea resurselor de la distanta de catre un calculator se realizeaza prin intermediul reţelei de calculatoare.

Implementarea unei retele de calculatoare în cadrul unei organizaţii prezintă o serie de beneficii, dintre care :

- partajarea informaţiilor,
- partajarea resurselor hard şi software, administrarea şi controlul centralizat.

Figura 1:replicarea bazei de date, date de o societate S

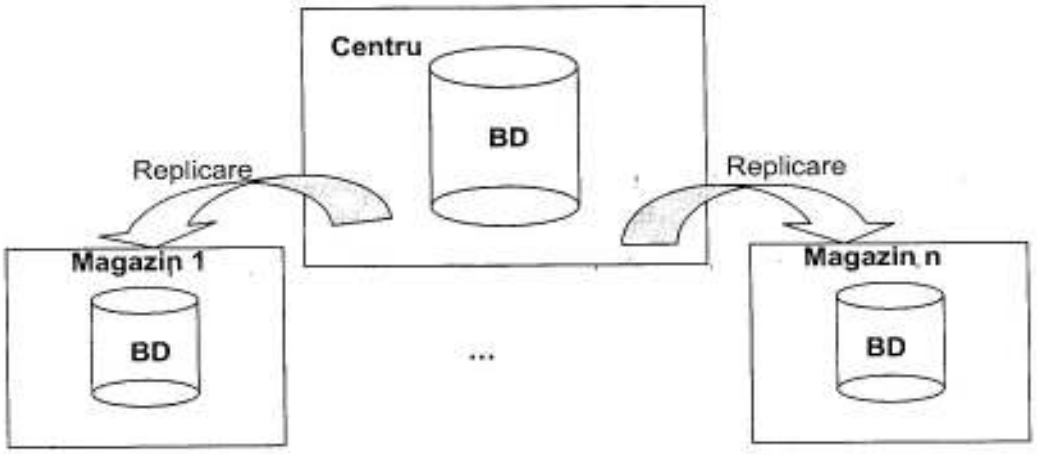


Fig.1 – Replicarea bazei de date a societății S

Implementarea unui server de baze de date care stocheaza datele şi transferă utilizatorilor raspunsul la interogarile solicitate.
Exemplu să consideram o societate S care drept domeniu de activitate *vanzarea de carti.*
Respectiva societate prin calitatea produselor oferite s-a impus pe piata de specialitate si a fost neviotă să se extinda prin deschiderea de noi magazine în oraşe diferite.
In aceasta situaţie există o singura baza de date care deţine date despre stocul exitent, clienti, furnizori, tranzactii, pretul produselor etc….
Aceasta baza de date este trimisă magazinelor, care la rîndul lor prelucreaza datele primite în funcţie de activitatea realizata.
Dupa cum reise din figura 1 fiecare magazin primeste o copie a bazei de date.
Procesul de replicare presupune interactiunea permanenta intre participantii la replicare.
In cazul cînd se modifică pretul unui produs, atunci baza centrala este modificata corespunzator.
Dupa reactualizarea bazei de date procesul de replicare îşi intra in drepturi, astfel, replica vechii baze de date este inlocuita cu replica baze de date actualizata in fiecare magazin.
Acest mod de lucru este foarte eficient deoarece fiecare magazin are cea mai recenta versiune a bazei de date centrale.

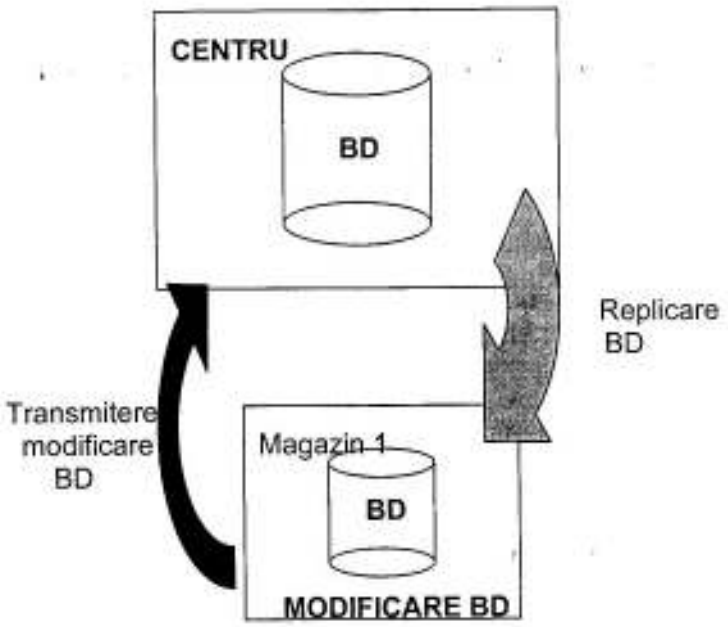


Fig. 2 – Replicarea în dublu sens

In figura 2 se prezinta procesul de replicare în dublu sens.
La nivelul fiecarui magazin apar modificari asupra continutului bazei de date, astfel este necesara pastrarea omogenitaţii datelor bazei de date.
In cazul cand se modifica baza de dat de la magazinul 1, este necesar ca aceasta modificare sa fie propagata in toate bazele de date.
Dupa figura 2, modificarea este transmisa bazei de date centrale care la randul ei transmite aceasta modificare tuturor celorlalte baze de date, în acest fel se reactualizeaza baza de date în toate magazinele.

2.Structuri de aplicatii distribuite

Pentru realizarea procesului de replicare este necesara implementarea serverului de replicare, care se ocupa cu gestionarea modificarilor care produc asupra bazelor de date

- este responsabil cu transmiterea şi recepţia modificarilor facute.

In cazul in care volumul de date este foarte mare ,respectiv modificari simultane de date, se realizeaza o coada de asteptare in care sunt puse respectivele modificari.

Clasificarea serverelor de replicare
1. server de replicare mai multi la mai mult.i
Serverul de replicare mai multi la multi este utilizat atunci cand numarul modificarilor este mic, un server de acest tip primeste de la mai multi indivizi toate modificarile produse pe care le transmite celorlalti membrii,in acest fel este reactualizata baza de date.

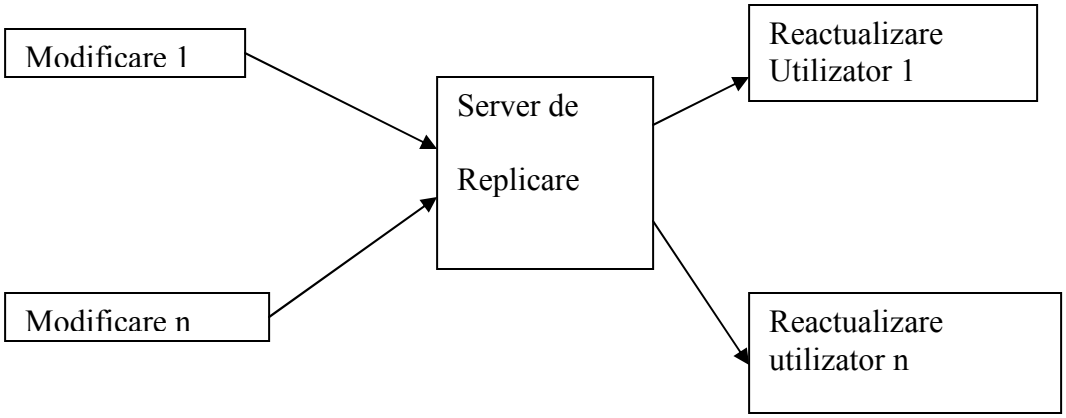


Figura 3. Server de replicare mai multi la multi

2.Tipul de server de replicare mai multi la unul este folosit când se doreşte centralizarea informaţiilor modificate.
Aceasta centralizare este logica,deoarece la momentul modificarii bazei de date de catre utilizator ceilalti membrii ai comunitatii de replicare lucreaza fizic doar pe prtiunea aferenta lui si nu cunoaste momentul transferului modificarii

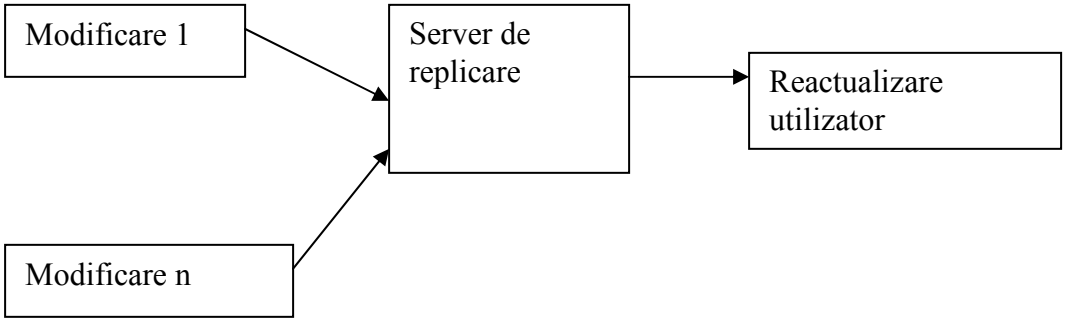
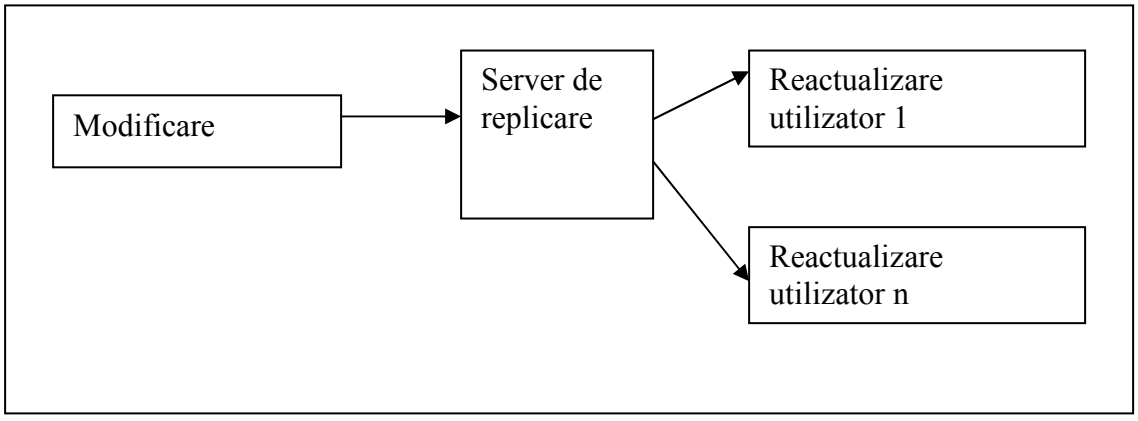


Figura 4 Server de replicare mai multi la unu

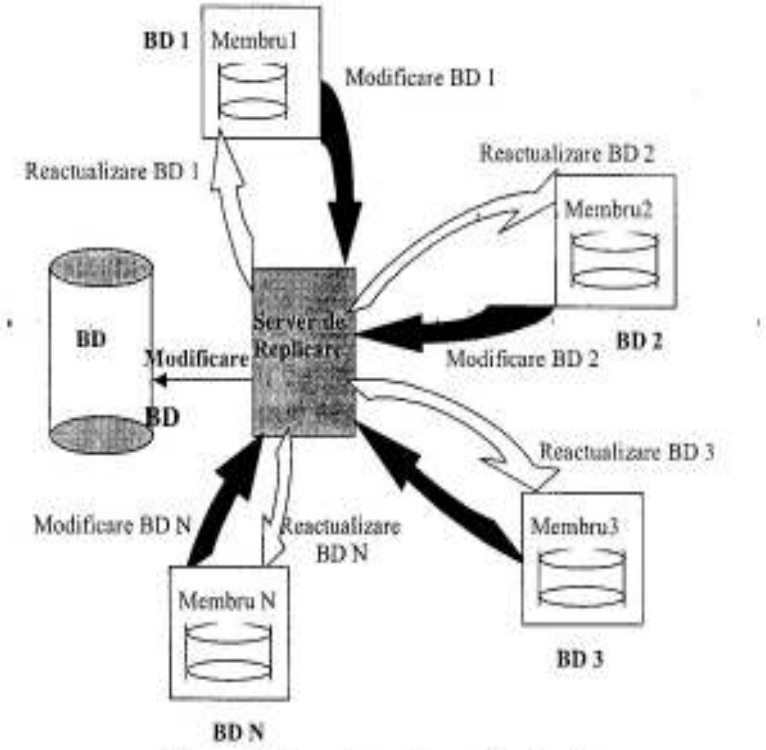
5.Server de replicare unu la mai multi
Acest server se foloseşte când se doreşte filtrarea modificarilor realizate dupa un anumit criteriu
Acest tip de server intercepteaza şi transmite numai acele tipuri de modificari care corespund filtrului impus.

Figura 5. Server de replicare unu la mai multi



Dupa cum se cunoaşte exista tipologiile linie, stea, cerc, hibrid.
Topologia unei reţele este data de modul de aranjare a calculatoarelor, a cablurilor, şi a altor componente din cadrul retelei, formând harta fizica a retelei.
Spre deosebire de reţea, topologia comunitaţii de replicare este data de locul unitaţii centralizatoare.
Societatea S are mai multe magazine in orase diferite de locatia sediului central, imaginea obtinuta este la fel ca si imaginea unei stele.
In cadrul acestui tip de replicare se realizeaza centralizarea logica a modificarilor produse in cadrul comunitatii de replicare, toate modificarile sunt trimise centrului care la randul sau le trimite membrilor comunitatii.
Acest tip de replicare este specific unitatilor subordonate unui departament aflat pe o treapta superioara in organigrama organizatiei.
Pentru departamentele de pe acelaşi nivel este specific tipului cerc, prezentat mai jos.

In figura 6. *este prezentata comunitatea de replicare de tip stea.*



Comunitatea de replicare de tip stea

Sageţile de culoare reprezintă modificarile operate de membrii comunităţii de replicare asupra bazelor de date locale.
Aceste modificari sunt trimise serverului de replicare care transfera modificarile catre baza de date centrala, precum şi celorlati membrii ai comunitaţii, reprezentate prin sagetile deschise.

Modelul în stea funcţioneaza cu un numar mai mic de servere de replicare decat numarul membrilor comunitatii.

Un alt model de replicare este modelul in cerc

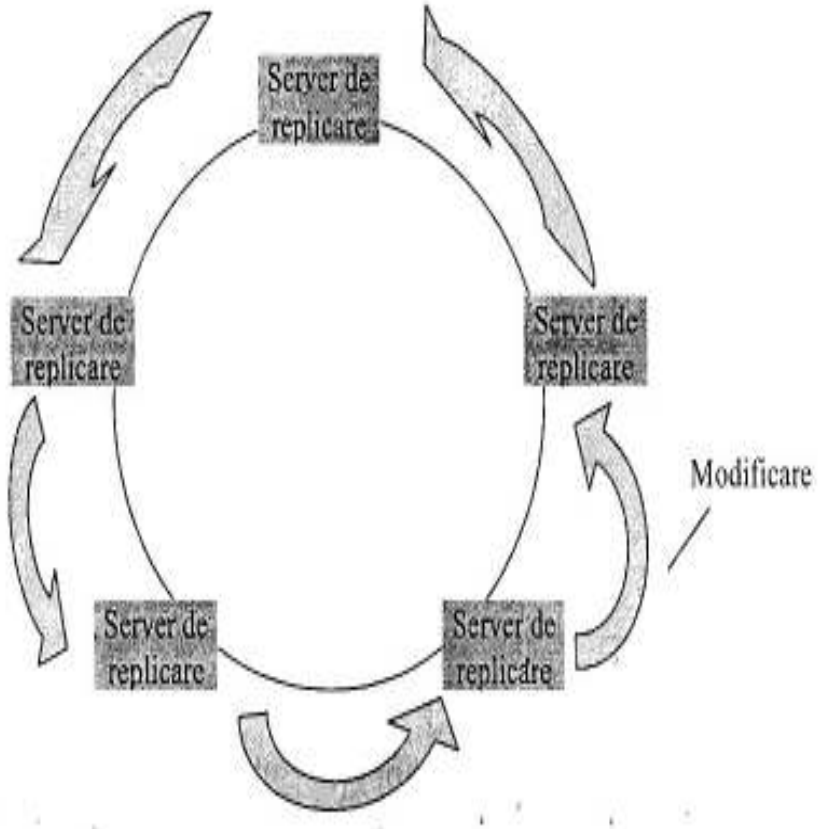
Acest tip de comunitate este specific departamentelor de pe acelasi nivel sau comunităţilor de replicare între ai caror membri exista legaturi de incrdere, neexistând necesitatea centralizarii modificarilor.

In acest caz sensul de transfer al modificarilor este unic şi anume invers arcelor de ceasornic

Un membru al comunitatii care opereaza o modificare, trimite respectiva modificare de serverului replicare propriu care redirectioneaza operatia catre serverul de replicare a urmatorului membru.

Acesta la randul sau o directioneaza spre urmatorul membru, procesul repetându-se pâna când in momentul revenirii la expediator.

Figura 7 Modelul de comunitate de replicare in cerc

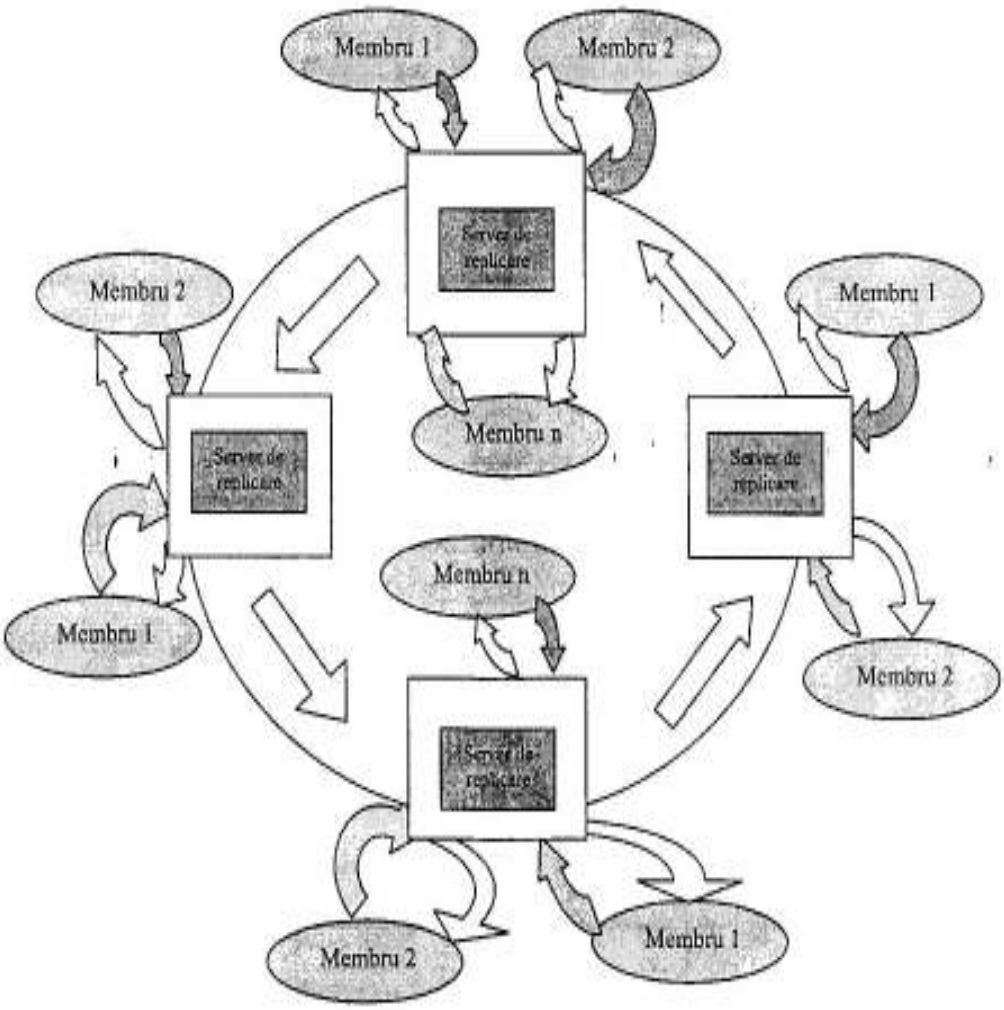


Modelul de comunitate de replicare în cerc

Acest model necesită implementarea unui numar important de servere de replicare, deci un numar cel putin egal cu numarul membrilor comunitatii.

Alegerea numarului de servere este determinata de numarul modificarilor simultane.

Un alt tip de model este cel hibrid, care utilizeaza celelalte doua modele prezentate anterior.
Model hibrid este util marilor organizatii unde este necesara comunicarea pe mai multe niveluri.
Figura 8 *Model hibrid de replicare*



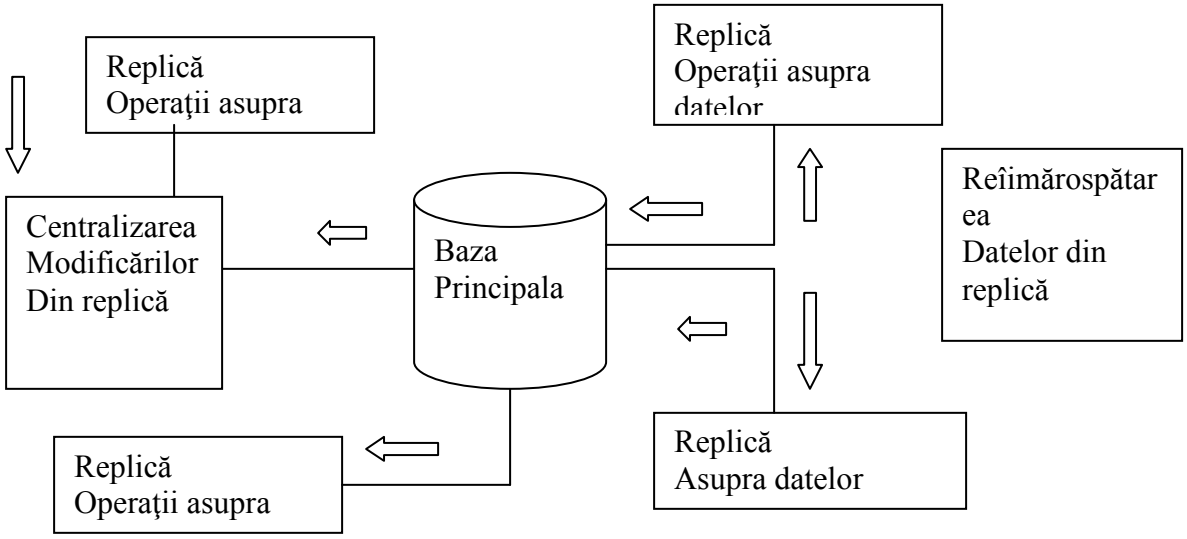
Modelul hibrid de replicare

3.Procesul de replicare

Proiectarea, gestionarea sau depanarea unui sistem de replicare presupune inţelegerea şi buna cunoastere atăt a imaginii de ansamblu dar şi a particularitaţilor ce definesc structura componenta în parte.
Replicarea bazelor de date este procesul prin care doua sau mai multe baze de date comunica intre ele, transferând date şi informaţii.
Acest proces stă la baza sistemului de replicare ce consta în unul sau mai multe servere care supervizeaza şi asigura transferul informaţional intre sursă si destinaţie.
La randul lor bazele de date sunt grupate în :

- baza de date principala care la orice moment reprezinta imaginea reala a intregului sistem si replici ale acesteia,care reprezinta imaginea reala a sistemului la momentul T, cand replica a fost creata.

Figura : 9
Asigurarea fluxului continu de date



Asemănarea cu sistemul bazat pe clone mai constă şi în existenţa modelului de *replicare parţial si total.*
In cazul partial se face replica doar la o parte din intreaga baza de date, astfel ca la momentul T, în unele noduri din sistem, se gasesc fragmente, iar in cazul replicare totala, toate nodurile conţin replica bazei de date centrale.
Sistemele de replicare utilizează tabele interne de replicare în care sunt inregistrate toate operaţiile care au avut loc asupra bazei de date, datele replicate fiind preluate din aceste tabele
Tipurile de replicare permise de serverele SQL, sunt de trei feluri :

1. statica ;
2. fuzionare ;
3. tranzacţionala
4. combinaţia lor

1.Replicarea statica este procesul de copiere şi distribuire a datelor exact aşa cum se află ele la un moment dat.

Acest tip de replicare nu necesita conectare permanentă pentru ca schimbarile de pe datele publicate nu sunt transmise incremental.
Replicarea va fi improspătată cu setul de date complet la a care a subscris fără a executa tranzacţiile individuale care au avut loc.
Replicarea statica necesită mai mult timp pentru propagarea datelor la un moment dat ce se replică tot setul de date replicarea are loc mai puţin decăt în cazul celorlalte tipuri de tranzactii, filtrarea datelor publicate, abonatii pot opera modificari asupra datelor primite, apoi pot propaga aceste schimbari.
Acest tip de replicare este util atunci când:

- Modificările asupra datelor nu sunt frecvente ;
- Este acceptabilă situaţia când datele sunt desincronizate pentru o perioada mai lunga de timp ;
- Datele replicate nu au un volum mare ;
- Site-urile sunt adesea deconectate ;
- Latenta mare este acceptabila.

2.**Replicarea tranzacţionala** presupune că o imagine iniţiala a datelor publicate este propagată la abonaţi, dupa care, când se fac modificari la replicator, tranzacţiile individuale sunt memorate şi reproduse la abonati, astfel incăt tranzacţiile sunt pastrate la abonat.
Acest tip de replicare este utilizabil când nu este tolerabila, o latenta mărita între bazele de date distribuite şi când se urmareşte in principal proprietatea de atomicitate.

3.**Replicarea prin fuzionare** permite site-urilor să funcţioneze autonom combinând modificările asupra datelor, uniformizând astfel continutul bazelor de date implicate în relatia replicator-abonat.
Actualizarile se fac în mod independent fără nici un protocol de realizare a tranzactiilor.
In mod natural pot aparea conflicte în actualizarea datelor , anumite servere furnizând mecanisme de rezolvare sau prin agenţi de fuzionare care , tipic, invocă proceduri stocate care compară datele şi decid care modificare se aplica general conflictelor.
Un s*istem de replicare a bazelor de date* respectă urmatoarele caracteristici distribuirii bazelor de date :

- *Atomicitatea* – orice acţiune asupra bazei de date se efectuează in totalitate sau nu se intămpla de loc ;
- *Consistenţa* bazei de date- plecând de la premiza ca baza de date se afla intr-o stare iniţială de consistenţa ;
- *Izolarea*- orice prelucrare sau alterare a unei zone din baza de date nu este replicata in intreg sistemul pâna când aceasta nu se incheie ;
- *Durabilitate*- orice modificare a datelor din baza de date care a fost transmisă tuturor parţilor sistemului de replicare este permanentă , fiind neafectată de caderea sistemului ;
- *Serializarea*- operaţiile asupra datelor din replici sunt preluate , realizate şi transmise altor replici de catre sistemul central de replicare

O alta caracteristica ,importantă a unui sistem de replicare de baze de date este modul în care sunt replicate modificarile efectuate asupra bazei de date centrale sau a unei replici.
Daca ele sunt replicate imediat ,atunci sistemul este sincron ,altfel este asincron

Primele limitari ale unui sistem sincron sunt de ordin fizic,de cele mai multe ori o reţea nu este operaţionala doar din simplu motiv al caderii legaturilor dintre diferite noduri ale sale sau funcţionarii imperfecte a acestora
În cazul sistemelor asincrone,toate aceste limitari impun reguli de pastrare a evidenţei acetor modificari fie pe serverul central de replicare,fie pe statiile unde se gasesc replicile,pâna la momentul in care se realizeaza replicarea completa a acestora in sistem

3.Softuri orientat pe replicare
Implementarea unui sistem de replicare de baze de date trebuie avuta în vedere urmatoarele :

- daca versiunea aplicatiei este compatibila cu tipul bazelor de date , în caz contrar existând probleme de portabilitate a datelor prin sistem şi chiar blocari
- testarea prealabila a aplicaţiei
- pe ce tip de replicare este bazata aplicatia statica, tranzactionala , dublu-sens, fuzionare şi ce tipuri de obiecte din bazele de date poate replica ;
- daca aplicatia are propriile rutine interne de depanare sau este necesară instruirea unui personal
- daca are facilităţi precum stabilirea de catre utilizator a propriilor reguli de replicare şi a mecanismului care declanşeaza replicarea.

Software orientat pe replicarea bazelor de date

Firma producatoare	Produs
Oracle (http://www.oracle.com)	Oracle 9i,10g
Sybase(www.sybase.com)	Sybase replication server 12.5
Microsoft(www.microsoft.com)	SQL Server 7.0
DataMirror(hhtp://datamirror.com)	Datamirror DB/XML Transform 2.1
Open Universal Software www.universal.com	Data distributor

c)Distribuirea mixtă

d)Distribuirea prin incarcare este tehnica de copiere periodica a intregii baze de date centralizate sau a unei porţiuni din ea pe noduri locale.

8.7.3 Distribuirea datelor din catalog

Funcţia de administrare a unui SGBD este extinsă datorită probelemelor care apar la distribuirea datelor, comparativ cu baza de date locala.
Astfel, daca la nivelul local sistemele nu au autonomie atunci administrarea distribuita nu se deosebeşte prea mult de administrarea centralizata normala.

a)Catalogul bazei de date distribuite

De exemplu folosind arhitectura binivel pentru un SGBDD, catalogul aferent bazei de date distribuite va conţine o serie de informaţii neceasre optimizarii acceselui la date şi asigurării integritatii şi securitatii datelor.
Aceaste informaţii se refera la schema globala, la fragmentare, la alocare, la accese si la protectia datelor.
Prezentăm îin continuare, cele mai importante informaţii care dau conţinutul unui astfel de catalog :
1.informaţiile despre schema globala :numele colectiilor globale, numele caracteristicilor (cămpurilor) din fiecare colectie ;

2.informatii despre fragmentare : calificarea fragmentelor (prin metoda orizontala,cămpurile din fragmente- metoa verticala, arborele de fragmentare prin calificare şi descrierea fragmentelor prin cămpuri, pentru metoda mixta.
3.informatii despre alocare : legaturile dintre fragmentele şi imaginile fizice ale colecţiilor globale, precum şi legaturile dintre imaginile fizice se datele memorate pe fiecare calculator din reţea.
4.informatii despre accesul la date : metadatele de acces utilizate pe fiecare calculator din reţea (index, hash, pointeri etc…), restrictiile de integritate, securitatea datelor
5.informatii statistice : indicatori statistici care dau profilul BD

Rolul catalogului bazei de date distribuite este de a furniza, în orice moment informaţii la zi despre starea bazei de date, utilizatorilor, a administratorului. Catalogul creat şi actualizat de SGBDD, el fiind accesbil utilizatorilor numai în citire.
Rolul catalogului este dat de urmatoarele aspecte :

- BDD este accesata de diferiţi utilizatori in cadrul anumitor aplicaţii.SGBDD asigura pentru aceştia diferiteniveluri de transparenta ajungand la datele fizice prin intermediul informatiilor din catalog, translatarea aplicatiilor
- SGBDD realizeaza o utilizare eficienta a datelor prin implementarea unor alogritmi de optimizare a alocarii şi accesului la date. Acesti algoritmi declanşaţi automat sau la cererea utilizatorului, folosesc informatiile de care au nevoie din catalogul bazei de date distribuite, optimizarea aplicatiilor
- Când un utilizator doreşte să acceseze baza de date in cadrul unei aplicatii, SGBDD verifica drepturile de acces si în caz favorabil, realizeaza conectarile corepunzatoare, executarea aplicatiilor.

b)Distribuirea catalogului
Analog cu funcţia de administrare a unui SGBDD, de care este strâns legat, catalogul bazei de date distribuite depinde de gradul de autonomie locala a calculatoarelor din reţea.
Avem că distribuirea catalogului depinde de autonomia locala şi atunci ea poate fi realizata în urmatoarele variante : replicat, local, centralizat, mixt.

b1)Catalogul replicat inseamna multiplicarea lui pe toate calculatoarele din reţea avantajul solutiei este reducerea de acces la consultarea catalogului , dezavantajul este cresterea spatiului de stocare şi a timpului consumat pentru actualizarea catalogului , deci trebuie actualizate toate copiile.

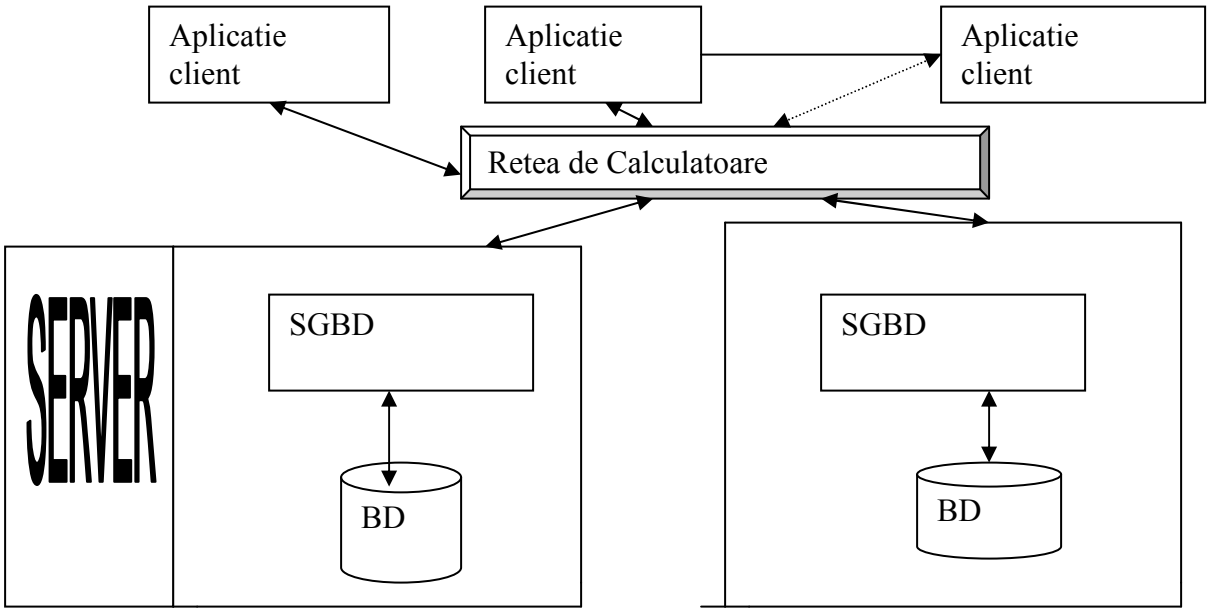
b2)Catalogul local presupune fragmentarea şi alocarea în acelasi mod cu datele din colecţia global ape care se refera, avantajul este că se reduce spatiul de stocare si timpul de actualizare, dezavantajul este că va creste timpul de consultare.

b3)Catalogul centralizat presupune alocarea lui pe un singur calculator , avantajul este că se reduce spatiul de stocare şi se simplifica implementarea, dezavantajul este că se complica concurenta accesului la catalog şi scade fiabilitatea sistemului.

b4)Catalogul mixt presupune combinarea a căte doua sau trei varaiante de mai sus

8.8 Exemple de SGBDD

Fig 8.8.1 Sistem de gestiune de baze de date distribuite



8.8.1 SGBDD omogene

In continuare prezentăm câteva sisteme omogene de baze de date distribuite bazate pe moelul relational extins, cu facilităţi de distribuire a datelor.

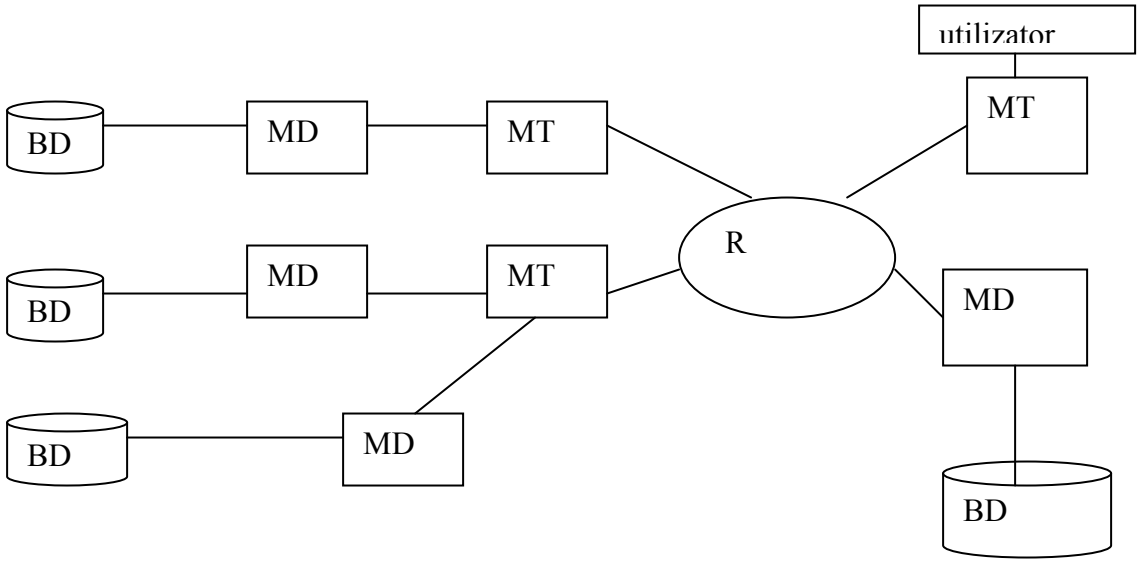
SDD-1 este primul ptotip de SGBDD dezvoltat de Computer Corporation of America

R* este dezvoltat de firma IBM .

INGRES distribuit este realizat la Unversitatea din California

POREL

SDD-1 Arhitectura



Arhitectura se bazeaza pe trei componente :
MD- Modelul de date care gestioneaza datele locale
MT- Modulul de tranzactie care planifica executia tranzactiilor
R- Componenta de reţea care interconectează modulele de date şi de tranzactie

Analiza comprativa a unor SGBDD omogene

In continuare, vom prezenta o comparatie între caracteristicele unor sisteme omogene de baze de date distribuite

Caracteristici	Sdd-1	R*	Ingres distribuit	Porel
Fragementare Orizontala Verticala mixta	Da Da Da	Nu Nu Nu	Da Nu Nu	Da Nu Nu
Transparenta fragmentarii	Da	TRANSPARTENTA	D	D
Intefete Interogare Limbaje gazda Precompilare	Da N 	D PL/I 	D C 	D Pascal D
Prelucrarea cererilor -cu o analiza initiala de alocare independenta	N N	D N	N N	 D
Separarea optimizarii globale fata de cea locala Costul btransmisiei	D D	N N	N N	D D

Folosind semijonctiunea	D	N	N	D
Blocarea in doua faze	N	D	D	D
Prevenirea blocarilor Descrierea blocarilor	D	N	N	N
Replicarea BD	D	N	D	D
Situatia globala a statiilor din retea	D	N	D	D
Structura de calcul Centralizata (C) Sau ierarhica (H)	C	H	C	C

8.8.2.SGBDD eterogene

Cele mai importante sisteme eterogene sunt : **MULTIBASE, DDTS.**

Multibase este un sistem pentru regăsirea datelor din bazele de date distribuite eterogene.

Principalele componente ale arhitecturii sunt :

- un sistem global de gestiune care asigura descompunerea unei cereri daplex în raport cu schema globala,
- o interfaţă cu baza de date locala care asigura translatarea unei cereri în LMD, execuţia cererii şi transmiterea rezultatului.

8.8.3.Cerinţe ale realizarii aplicaţilor distribuite

Aplicaţii informatice distribuite

Un sistem de gestiune a bazelor de date distribuit (SGBDD) omogen când componentele din care este construit (componente hardware si software)sunt de acelasi fel.

Un sistem distribuit este neomogen (etrogen), a caror componente hradware şi software sunt diferite, care sunt realizate in diverse limbaje de programare (Visual Foxpro (VFP) şi ORACLE), respectiv componente conceptuale (topologia retelor care intra in alcatuirea sistemului distribuit, modul de counicare, sincronizare şi coordonare intre procese etc…) sunt diferite.

Necesitatea proiectarii unor sisteme informatice distribuite este motivată de aparitia unor avantaje.

Cele mai importante dintre aceste avantaje sunt :

- Schimbul de informaţii-cresterea masivă a cantităţii de informatie şi necesitatea de a schimba rapid informatii intre diferitele puncte aflate in locuri geografic departate care fac necesara conectarea intre calculatoare autonome
- Partajarea resurselor
- Siguranta marită in funcţionare
- Performanţe mărite
- Specializarea nodurilor- proiectarea unui sistem de calcul autonom, cu mai multe functionalitati, utilizarea unor algoritmi in sistemele distribuite.

APITOLUL 9
Aplicaţii Visual Foxpro (VFP) cu servere de baze de date ORACLE

9.1.Specificul aplicaţiilor cu servere de baze de date Oracle [8]

Sistemele informaţionale economice sunt construite de regula în jurul unui server de baze de date, care îin acest mod sunt disponibile în medii client/server sau pe web.

Bazele de date distribuite implementate în astfel de medii sunt proiectate respectând cu stricteţe anumite canoane, privind modurile de organizare şi sunt accesbile prin limbaje şi structuri conforme.

Caracteristicele generale ale unor astfel de sisteme pot fi sintetizate dupa cum urmează :

- Protecţia datelor impotriva unor prejudicii **(arhivarea şi recuperarea-backup si recovery)**
- Asigurarea accesului la date a unui numar mai mare de urtilizatori **(concurenta),** respectiv mai multor aplicaţii
- Asigurarea integrităţii datelor, prin reguli stricte- **restricţia de integritate referenţiala**
- Interzicerea accesului neautorizat la date (**securitatea datelor)**

Securitatea BD reprezintă protecţia impotriva accesului sau modificarilor neautorizate

accesul la resursele sistemului poate avea loc prin facilităţi de natura , paralolelor, profilelor utilizator sau matrice ale drepturilor de acces (privilegii)

- Izolarea detaliilor referitoare la manipularea datelor pe diferite platforme **(portabilitatea)**

*Portabilitatea s*oftware-ului este capacitatea acestuia de a se executa pe diferite platforme hardware.

In cazul aplicaţiilor cu baze de date problema portabilităţii are doua aspecte :

- portabilitatea la nivelul SGBDD
- portabilitatea la nivelul sistemului de operare(S.O).

9.1.1 Arhivarea şi Recuperarea -Oracle backup &recovery

Arhivarea şi recuperarea (backup & recovery) reprezintă unul dintre cele mai importante aspecte ale administrării bazei de date.

Singura cale de a asigura recuperarea bazei de date în timp util, fără pierderi de date este stabilirea unui plan şi utilizarea acesteia.

A.arhivarea bazei de date

Exista trei tipuri de arhivare a bazei de date: **offline, online, logic**

Arhivarea bazei de date include toate fişierele externe ale unei instanţe(din directoul *%ORACLE_HOME%/Database)*

*Problema este ce trebuie să arhivam ? Î*n continuare trebuie să vedem care sunt fişierele externe cu care lucreaza Oracle:

1. *fisierul de parametrii (<init<SID>.ora)* conţine parametrii de iniţializare şi configurare a instanţei Oracle

2 .*fişierul jurnal(redo log)*-conţine informaţii privind tranzacţiile sau modificările operate în BD, aceste informaţii sunt necesare la recuperarea conţinutului bazei de date.

3. *fişierul jurnal arhivat(archive logs)*-contine copii ale fisierelor jurnal necesare pentru recuperarea bazei de date, aceste apar atunci când baza de date ruleaza în modul ARCHIVELOG
4. *fişierul de control (control file)* –conţine informaţii pentru pornirea şi funcţionarea bazei de date
5. *fişierul de parole(password file)*-conţine informaţii privind îndentificarea utilizatorilor privilegiaţi
6. *fişierele de date*- contine informatii despre baza de date şi dicţionary
7. *t ablespace-urile* –sunt formate dintr-un set de segmente şi sunt utilizate pentru a grupa logic obiectele din baza de date şi pentru a defini modelul de alocare a spatiului pe disc.

Tablespace-ul System – conţine dicţionarul bazei de date şi segmentele de revenire , system rollback segment).
Pentru re-crearea segmentelor de index trebuie să avem instructiunea de creare, *create index.*
Segmentele de revenire care nu conţin tranzacţii pot fi refacute în acelaşi mod , *create rollback.*

Tablespce-ul asigura corespondenţa între obiectele bazei de date la nivelul logic şi structura fizică a BD.
Un tablespace conţine unul sau mai multe fişiere de date care trebuie arhivate
Numele şi locatia fisierelor de date pot fi regasite folosind coloana name din vederea

V$DATAFILE, interogarea:
SELECT NAME FROM V$DATAFILE;
Rezultatul interogarii este:
C:\ORANT\DATABASE\SYSIORCL.ORA
ETC……
Membrii fişierelor jurnal pot fi obţinuti din view-ul *v$logfile* cu introgarea

SELECT name FROM v$logfile; reazultatul este:
C:\orant\database\log10rcl.ora
C:\orant\database\log20rcl.ora
Locatia fiserelor de control poat fi regasita utilizând coloana name din view-ul v$CONTROLFILE cu interogarea
Sql>select name from v$controlfile;
Reazultatul este:
C:\orant\database\ctl10rcl.ora
C:\orant\database\ctl20rcl.ora
D:\backup\ctl30rcl.ora

Modurile ARCHIVELG şi NOARCHIVELOG
Fişierele jurnal inregistrează toate tranzacţiile active din baza de date. Pentru a conserva spatiul pe disc Oracle reutiliezaza fişierele jurnal prin scrierea lor în mod circular.
Astfel , când toate fişierele jurnal al bazei de date sunt pline, procesul background LGWR va începe să rescrie primul fişier jurnal.
Daca baza de date ruleaza în modul NOARCHIVELOG , tranzacţiile din fişierele jurnal se vor pierde datorită rescrierii acestor fişiere, dar numai arhivarea *offline* este posibilă.
Daca baza de date ruleaza in modul ARCHVELOG, procesul background ARCH va copia fişierele jurnal de pe disc îin spatial de arhivare al BD.

Pentru a comuta între *NOARCHIVELOG- ARCHIVELOG* se parcurg următorii pasi:

1. *oprirea bazei de date* in modul NORMAL sau IMMEDIATE
2. *Se editeaza fisierul init<sid>.ora* şi se adauga tre noi parametrii:
    - Log_archive_start=true=pornirea arhivarii automate
    - Log_archive_dest=/disk_device/archive_id=destinaţia fişerului jurnal
    - Log_archive_format=%s.log =formatul numelui fisierului jurnal arhivat

3. pornirea bazei de date cu comanda START UP MOUNT

4. pornirea modului ARCHIVELOG

ALTER DATABASE ARCHIVELGŞ

*5.inchidera bazei de date :***alter database open**

În modul ARCHIVELOG se poate opta între arhivarea automata sau manuala a fişerelor jurnal controlat de parametrul *log_archive_start.*
Alter system archive log start: = pornirea arhivarii automate fără a opri instanţa.
Pentru a determina modul de arhivare a fişierelor jurnal se utilizeaza comanda : a*rchive log list.*

Arhivarea offline este considerată o arhivare consistentă, deoarece toate blocurile bazei de date corespund unui anumit moment în timp.
Arhivarea offiline presupune oprirea instanţei cu o comanda de oprire normală şi apoi copierea fişierelor externe.
Optiunea abort va solicita ca procesul SMON să execute recuperarea automata la repornirea bazei de date.
Deoarece recuperarea automata nu garanteaza refacerea bazei de date, nu se recamnada arhivarea offline dupa oprirea su SHUTDOWN ABORT
Este indicat ca fişierul de control, să fie generat intr-un fisier text:
Alter database backup controlfile to 'c:\orant\database\control.txt';

Arhivarea online este considerată o arhivare inconsistenta , care presupune modul ARCHIVELOG şi copierea fisierelor de date, fişierul de control şi jurnal arhivate.
Arhivarea online incepe cu comanda ARCHIVE LOG LIST
Procesul de arhivare online presupune arhivarea fisierelor de date asociate fiecarei tabele de spatiu.
Paşi pentru arhivare unui tablespace
1)comanda Oracle pentru a porni arhivarea unei tabele de spatiu
Alter tablespace tablespace_name begin backup:
2*)comanda* de copiere a sistemului de operare pentru salvarea fişerelor de date associate tabelei de spatiu:
Alter tablspace tablespace_name end backup;
3) Utimul pas este arhivarea fisierului de control
Alter database backup controlfile to 'c:\orant\database/control.txt'

Cei trei paşi trebuie repetati pentru fiecare tabela de spatiu a unei instante, iar dupa arhivarea tuturor se foloseşte comanda *: archive log list* pentru a vedea care este fişierul jurnal curent la terminarea arhivării.

Arhivarea logica (EXPORT)
Arhivaea logica se utilizeaza numai pe o baza de date online mica, cu puşine tranzacţii
Figura 1 script de creare a programului de arhivare online

Arhivare.sql

```
**********************************************
Rem descriere acest script realizeaza arhivarea online a fisierelor de date
corerspunzatoare tabelelor spatiu si a fis.de control.
Rem fisier de iesire arhivareonline.sql

Create or replace procedure arhivare(unde_sa_arhivez in varchar2)
Is
Fname    varchar2(80);
Tname varchar2(80);
Tname1 varchar2(80);

Cursor cursor1 is
    Select tablespace_name,file_name
     From v$datafile.sys.dba_data_files
     Where enabled like '%write%'
And file#=file_id
Order by 1;

Begin
Dbms_output.enabled(3200);
Dbms_output.put_line('rem arhivarea online a tuturor fisierelor ori date in
directorul
'|| unde_sa_arhivez);
Dbms_output.put_line('**************************');

If cursor1%isopen
Then
Close cursor1;
End if:

Open cursor1;
Fetch cursor1 into tname,fname:
Tname1:=tname:
Dbms_output_line('alter tablespace '||tname'begin backup:');

While cursor1%found loop
   If tname1 !=tname then
Dbms_output.put_line('alter tablespace '||tname||' end backup:');
Dbms_output.put_line('**********************');
 Dbms_output.put_line('alter tablespace '||tname||' end backup:');
Tname1:=tname:
End if;
Dbms_output.put_line('rem arhivarea online a tuturor fisierelor ori date in
directorul
'|| unde_sa_arhivez);
Fetch cursor1 into tname,fname;
End loop;
Dbms_output.put_line('alter tablespace '||tname||' end backup:');

Close cursor1;
Dbms_output.put_line('************************');
Dbms_output.put_line('rem arhivarea fis.de control');
```

```
Dbms_output.put_ine('alter database backup controlfile to trace:');
Dbms_output.put_line('alter database backup controlfile to '||''''||
            Unde_sa_arhivez||'/control.'||
To_char(sysdate,'DDMMYYYYHH24MISS')||''''||':');
END;
/
Rem executarea proceduri va produce un fisier text cu comenzi online.sql

Set serveroutput on
Set heading off
Set feedback off
Spool online.sql

Execute arhivare('c:/backup');

Spool off
Set heading on
Set feedback on
Set serveroutput off
```

Strategii de recuperare a bazelor de date Oracle

Principala responsabilitate a administratorului bazei de date este să fie pregatit din timp în cazul aparitiei unei căderi hardware, software, de retea, de process sau de sistem.

Tipuri de eşec

1)esecul instantei(Database Instance Failure)-apare când un eveniment impiedica rularea normala a proceslor background (**PMON, SMON, DBWR, LGWR**).Acest tip de eşc poat fi cauzat de o problema hardware sau software.

Realizarea instantei se realizeaza automat, procedura implcând doua faze: d*erularea inanite a* tranzactiilor comise aplicand fisierele jurnal online si derularea inapoi a tranzacţiilor necomise utilizand segmentele de revenire Aceasta procedura se executa si atunci cand baza de date este repornita dupa inchiderea ei cu **ABORT-SHUTDOWN**

2)Eşecul media de disc (media disk failure)-se manifesta prin incapacitatea BD de a citi sau de a scrie date , iar cauzele include defectarea discului, blocuri etc..

Recuperarea media se realizeaza prin restaurarea fisierelor pierdute dintr-un backup recent şi apoi aplicarea fisierelor **log online** sau arhiva .

Tipuri de recuperări

- *tipuri de recuperare media*
- *.pornirea bazei de date*
- *.recuperarea la nivelul bazei de date*
- *.recuperarea la nivelului tabeluli*

Comanda RECOVER ,are formatul general :

RECOVER[AUTOMATIC][FROM **'arhive_file_location']**
|[STANDBY][DATABASE[UNTIL CANCEL]|[UNTIL TIME time]|[UNTIL CHANGE integer]
[using backup controlfile]
Tablespace 'TABELA_SPACE_NAME'
Datafile 'DATAFILE_NAME'
Continue

Cancel

Recuperarea la nivelul bazei de date-se utilzeaza când se recupereaza tabela de spatiu SYSTEM, un fisier de date ce face parte din tabela spatiu SYSTEM sau contine segmente de revenire active sau când se executa orice tip de recuperare incompleta.
Optiunea UNTIL CANCEL –realizează recuperare incompletă şi se utilizeaza când un fişier jurnal arhiva este deteriorat Avem

1. restaurarea fisierelor de date si a fisierelor jurnal arhiva din cel mai recent backup
2. montarea BD cu comanda STARTUP MOUNT din SERVER MANAGER
3. Comanda RECOVER DATABASE UNTIL CANCEL-produce recuperarea BD aplicand primul fisier jurnal din secventa si ofera posibilitatea recuperarii anularii recuperarii inante de a aplica urmatorul fisier jurnal
4. comanda RECOVER DATABASE CANCEL- produce oprirea procesului de recuperare
5. Deschiderea BD cu comanda ALTER DATABAE OPEN RESTLOGS

Opţiunea UNTIL TIME- relizează o recuperare incompletă

1. restaurarea fişierelor de date şi a fisierelor jurnal arhiva din cel mai recent backup
2. montarea BD
3. Comanda *RECOVER[AUTOMATIC] DATABASE UNTIL TIME '2006-04-24 15 :59 :59'*
4. Deschiderea BD

Optiunea *UNTIL CHANGE*- realizeaza o recuzperare incompletă, pâna la un moment dat SCN(SYSTEM NUMBER CHANCGE).Pentru a se vedea SCN-ul tranzacţiilor comise se pot folosi view-urilor **system v$log_history)fisierul jurnal arhiva) si v$log (jurnal online)**
Select archive_name,low_change#,high_change# from v$log_history;
Select group#,members,first_change# from v$log;

Recuperarea la nivelul tabelei de spatiu- este o recuperare consistenta, deoarece se aplica la toate fisierele arhiva, în acest caz BD trebuie deschisa, iar tabela spatiu să fie offline

1. Deschiderea bazei de date
2. Comanda ALTER TABLESPACE nume_tab OFFLINE IMEDIATE SELECT * FROM V$DATAFILE

1. Restaurarea fişierelor de date corespunzatoare tabelei de spatiu
2. Comanda Recover [AUTOMATIC]TABLESPACE nume_tab
3. se aplica comanda ALTER TABLESPACE nume_tab ONLINE

Tabele de spatiu ce contin segmente temporare, de index sau segmente de revenire pot fi refacute prin stergerea tabelei de spatiu cu comanda DROP TABLESPACE nume_tab, stergerea fizica de pe disc a fisierelor de date corespunzatoare acesteia, refacerea tabelei de spatiu şi a segmentelor, folosind comenzile :**CREATE TABLESPACE,CREATE INDEX,CREATE ROLLBACK**

Recuperarea la nivelul fisierului de date- este o recuperare consistentă, asemanatoare recuperării la nivelul tabelei de spatiu

1. deschiderea bd
2. comanda *alter database nume_fis offline*
   Restaurarea fisierului de datre din cel mai recent backup

1. comanda recover[automatic] datafile nume_fis
2. ALTER DATABASE nume_fis offline

Deteriorarea tuturor fişierelor online dintr-un grup care nu este activ
Când procesul LGWR va incearca să scrie in membrii unui grup redo log deteriorat, automat toti utilizatori BD vor fi deconectati, prin mesajul **ORA-01092:ORACLE instance terminated.Discoonection forced**, daca se incearca o noua conexiune la BD se primeste mesajul de roare:**ORA-03114 not connected to ORACLE sau ORA-00472:pmon process terminaned with error**
Etapele de realizare:

1. Redeschiderea BD cu comanda *STARTUP FORCE (SHUTDOWN ABORT)*
2. Interogarea :select bytes/1024 from v$log where group#=3;- pentru a vedea marimea in K a fisierelor jurnal pierdute
3. stergerea grupului redo log 3-alter database drop log-file group 3;
4. (recuperarea grupului redo log 3:alter database add logfile group 3('/cale/log31.log','/cale/log32.log')size 500k;
   Deschiderea bazei de date

Deteriorarea tuturor fişierelor jurnal online dintr-un grup care este activ:
In acest caz apare mesajul "**ORA-00257:ARCHIVER ERROR……**….
Etapele de realizare sunt :

1. montarea bazei de date cu comanda STARTUP MOUNT din SERVER MANAGER
2. Interogarea :select group#,sequence#,bytes,first_change#,first_time,status from v$log;

Concluzii
1) *Cum determinam ce eşec a aparut ?* Multe dintre problemele aparute pot fi monitorizate din fisierul de alerta (ALERT LOG), ce contine informatii generale despre pornirea , oprirea bd si erorile intalnite pe parcursul functionarii acesteia şi prin consultarea fisierelor de urmarire (TRACE FILES), ce conţin informatii despre proces de background in parte (DBWR,LGWR,ARCH,PMON ETC…)
Fisierele de alerta şi fisierele de urmarire se gasesc intr-un director controlat de parametrul de intializare **BACKGROUND_DUMP_DEST**
2)Cum determinam ce trebuie sa recuperam ? putem detecta fisierele ce vor fi recuperate prin interogarea view-urilor de perfomanţa sistem V$:

- V$DATAFILE- informatii despre fisierele de date ;
- v$logfile,v$log- informatii despre fis.jurnal online ;
- v$log_history-fisierul arhiva ;
- v$recover_file-fisierele care necesita recuperare

3)perfomanţa şi succesul planului de arhivare şi recuperare depinde de doi factori: timpul necesar realizarii procesului de recuperare şi pierderile de date aparute.

.

9.1.2.Concurenţa este proprietatea bazei de date de a fi accesată de mai multe procese în acelaşi timp, fie acestea aplicaţii care se executa local sau la distanţa.
Gestiunea tranzacţiilor [3]
SGBDD lucreaza cu mai mulţi utilizatori, care acceseaza concurent datele din tabele, accesul concurent al utilizatorilor este asigurat prin capacitatea de multiprogramare a S.O, care permite executia concurenta a mai multor procese.
A.Tranzacţii
Definitie

O tranzacţie este o unitate logica de prelucrare indivizibila (atomica), a datelor unei baze de date prin care se asigura consisţenta acesteia.

***A.1.Anomalii* de execuţie concurenţa a tranzacţiilor**

Pentru studierea controlului concurentei şi al refacerii datelor se vor aborda operatiile asupra bazei de date la nivel de articol de date şi bloc de memorare pe disc.

A.2.Proprietăţile tranzacţiilor[3]

Cele mai importante proprietăţi ale tranzactiilor sunt identificate astfel **ACID (atomicitate,consistenta,izolare,durabilitate)**

***a)Atomicitatea* e**ste proprietatea unei tranzacţtii de a reprezenta o unitate de executie indivizibilă adica de a executa totul sau nimic.

b)consistenta unei tranzactii inseamna proprietatea acesteia de a efectua modificari corecte ale bazei de date, sau o tranzactie transforma o BD dintr-o stare consistenta în alta stare consistenta.

c)izolarea este proprietatea unei tranzactii de a face vizibile modificarile efectuate dupa ce a fost validata (commit) daca in acest timp sunt executate alte tranzactii concurente, acestea nu văd modificarile partiale efectuate de tranzactia respectiva pâna in momentul validarii tranzactiei.

d)durabilitatea este proprietatea prin care, dupa validarea unei tranzactii, modificarile efectuate de acestea in baza de date nu vor mai fi pierdute datorita unor defectari de sistem.

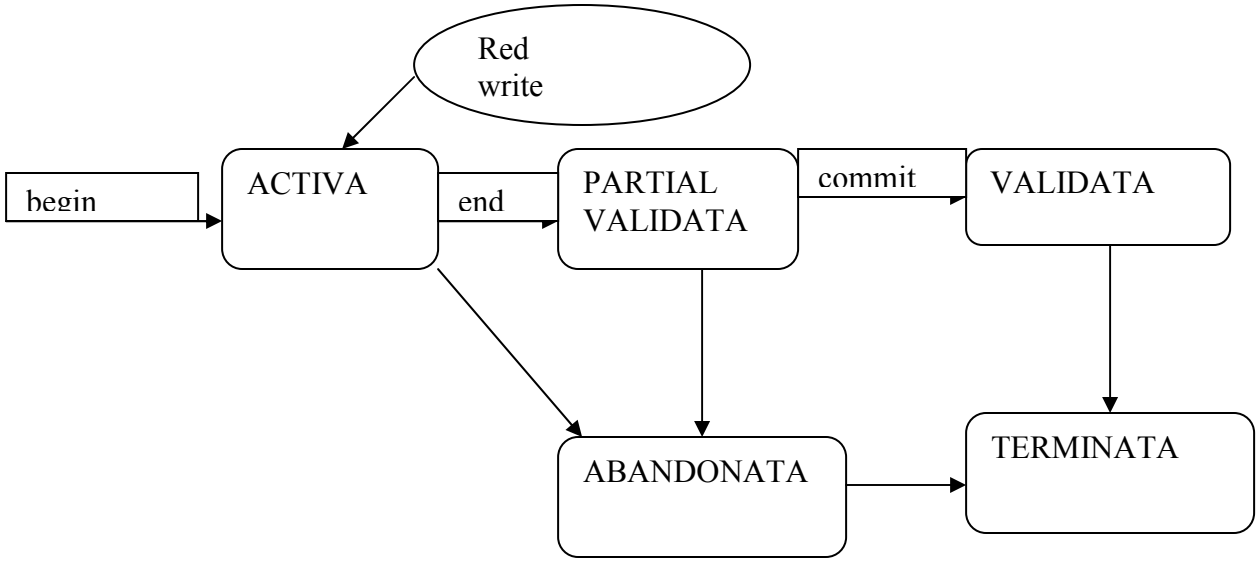
A.3.Stările tranzactiilor[3]

In continuare se va studia cazul general, al tranzacţiilor de citire şi scriere care efectueaza atât regasiri cât şi actualizari , care necesita tehnici de control al concurentei mult mai complexe.

Operaţiile efectuate de o tranzacţie şi inregistrate de administratorul de *refacere(recovery* manager), sunt urmatoarele :

1. begin : aceasta operatie marcheaza inceputul executiei unei tranzactii
2. read sau write :operatii de citire si scriee
3. end
4. commit : aceasta operatie semnaleaza terminarea cu success a tranzactiei
5. rollback (abort) aceasta operatie semnaleaza că tranzactia respectivă a fost abandonată
6. undo este similara operatiei rollback, dar se aplica unei singure operatii
7. redo: aceasta operatie specifica faptul ca unele operatii trebuie sa fie executate di nou pentru a se putea valida intreaga tranzactie

Figura diagrama de stare a unei tranazctii



A.4 Planificarea tranzacţiilor[3]

O planificare S a n tranzactii T1, T2,….Tn este o ordonare a operatiilor tranzacţiilor astfel incât, pentru orice tranzactie Ti care participa in S ,operatiile lui Ti în S respectă ordnea iniţială din Ti.

A.5 Serializabilitatea planificarilor[3]

Daca doi utilizatori ai unei baze de date lanseaza simultan doua tranzactii T1 şi T2 şi, nu este permisa nici o intreţesere intre operaţiile celor doua tranzactii, atunci sunt posibile doua moduri de ordonare a operatiilor celor doua tranzacţii

1. se executa mai inati toate operaţiile tranzacţiei T1, urmate de executia tuturor operatiilor tranzactiei T2
2. se executa mai inati toate operatiile tranzactiei T2, urmate de executia tuturor operatiilor tranzactiei T1

Astfel de planificari, care nu intreţes operatiile tranzactiilor, ci le executa consecutiv, se numesc **planificari seriale**

Definţie
Planficari seriale

O planificare S se numeste seriala daca pentru orice tranzacţie T participantă in planificare, toate operatiile lui T se executa consecutiv in S, astfel, planificarea se numeşte **neseriala**

Exemplu

S_A r1(X) ;W1(X) ;R1(Y) ;W1(Y) ;C1 ;R2(X) ;W2(X) ;C2 ;
S_B :R2(X) ;W2(X) ;C2 ;R1(X) ;W1(X) ;R1(Y) ;W1(Y) ;C1 ;

Definiţie
Planificari serializabile

O planificare a n tranzactii se numeste serializabila daca este echivalenta cu o planificare seriala a celor n tranzacţii

Def
Planificari echivalente din p.v al conflictelor

Doua planificari sunt echivalente din pnct de vedre al conflictelor daca oricare pereche de operatii conflictuale se executa in aceeasi ordine in cele doua planificari

B Tehnici de control al concurenţei

B.1Controlul executiei concurente a mai multor tranzactii este necesar pentru a asigura proprietatile ACID,si prin consistenta datelor

Controlul concurentei se poate realiza prin protocoale (set de reguli) impuse de tranzactii.

Cele mai importante tehnici de control al concurentei sunt cele bazate pe blocarea datelor prin intermediul **zăvoarelor (locks)** şi cele bazate pe **marci de timp**

1.Controlul concurentei prin blocare se face folosind zăvoarele

Un zăvor este o variabila asociata cu un articol al unei BD care descrie starea acelui articol in raport cu operatiile care se pot aplica acelui articol.

B.1Tipuri de zăvoare

In SGBD se pot utiliza doua tipuri de zavoare : zaboare binare şi cu stari multiple

Def :

Un zavor binar poate ave adoua stari : liber (sau neblocat-unlocked) şi ocupat (sau blocat-locked) sau, simplu , starile 1 si 0

Asupra unui zavor L(X0 se pot executa doua operaţii : operatia de blocare :lock(x) şi de eliberare unlock(x)

Algoritmul
Operatia lock(x)
Bucla :TS L
JZ BUCLA
Zavorul L este ocupat
Executie operatii asupra articolului X
Operatia unlock(x)
STORE L,1

In cele ce ce urmeaza se pot peciza urmatoarele reguli :

1. O tranzactie T trebuie să lanseze o operatie de blocare a zavorului asignat articolului X (lock(x), inainte de a efectua operatia de citire, read(x) sau de scriere write(x)
2. O tranzactie T trebuie sa elibereze zăvorul unui articol X prin operatia unlock(x), după ce a efectuat operatiile de citire si scriere
3. O tranzactie T nu poate cere un zavor pe care il detine deja
4. O tranzactie T nu poate elibera un zavor pe care nu il detine

B.2Protocolul de blocare in doua faze[3]

O tranzactie respecta protocolul de blocare in doua faze daca toate operatiile de blocare a zăvoarelor preced prima operatie de eliberare a unui zavor

B.3 Impasul si amanarea nedefinita

Impasul este o blocare a executiei tranzacţiilor care apare atunci când doua sau mai multe tranzacţii se aşteapta una pe cealalta ca să elibereze un zăvor.

Pentru rezolvarea impasului se pot folosi doua soluţii :prevenirea impasului sau detecţia şi eliminarea impasului.

Prevenirea impasului se poate face prin protocoale care impiedica aparitia acestuia, dar limiteaza gradul de concurenta al executiei tranzactiilor

O solutie pentru a rezolva problema impasului consta in detectia acestuia dupa ce a avut loc si eliminarea lui prin abandonare uneia din tranzactiile participante si anularea (rollback) a operatiilor acesteia.

Pentru detectia impasului se construieste un graf de aşteptare sau graf de percedenta ale carui noduri reprezinta tranzactiile aflate in executie curentă

Def :

Amănare nedefinită

O tranzactie se afla în stare de amănare nedefinita daca ea nu poate continua executia pe o peroiada lunga de timp, în timp ce toate celelalte tranzactii se execută normal.

Solutia standard de rezolvare este de a asigura o schema echilibrata de aşteptare a obţinerii zăvoarelor.

O astfel de schema este o coada de aşteptare de primul venit, primul servit (FIFO), în care tranzacţiile au posibilitatea de a bloca un zăvor în ordinea în care ele au cerut blocarea acestuia.

C. CONTROLUL CONCURENTEI BAZAT PE MARCI DE TIMP

Def :O marca de timp este un identificator unic al unei tranzactii, creat de SGBD, care se bazeaza pe timpul de start al tranzacţiei.

O marca de timp se poate creea fie folosind valoarea curenta a ceasului sistemului, fie folosind un numarator care este incrementat la fiecare asignare a unei noi marci.

Pentru fiecare articol X al BD sse definesc doua marci de timp :

1. read_TS(X)- marca de timp de citire
2. write_TS(X)- marca de timp de scriere

C.1 Controlul concurentei bazat pe ordonarea operatiilor upa marcile de timp[3]

In tehnica de control al concurenţei bazat pe ordonarea operatiilor dupa marcile de timp , ori de căte ori o tranzactie T încearca să citeasca să scrie in artocolul X , se compara marca ei de timp cu mărcile de timp de citire şi de scriere.

La lanasera unei operaţiilor de scriere a articolului X (write(x)), poată să apară urmatoarele situaţii :

a)daca read_TS(X)>TS(T) , atunci tranzacţia T trebuie să fie abandonată şi rulată înapoi

b)daca write_TS(X)>TS(T), atunci tranzactia T nu va executa operatia de scriere

c)daca nici una din situatele de mai sus nu au aparut, atunci T va executa operatia de scriere in X şi va seta WRITE_TS(X)=TS(T)

d)daca write_TS(X)>TS(T), atunci tranzactia T trebuie să fie abandonată şi reluata îinapoi

e)daca write(x)_TS(X)<=TS(T), atunci tranzacţia T va executa operatia de citire din articolul X şi va seta marca read_TS(X).

C.2 Controlul concurentei bazat pe versiuni multiple ale articolelor[3]

Controlul concurent bazat pe versiuni multiple ale articolelor foloseşte toate mărcile de timp ale tranzacţiilor şi ale articolelor.

Fiind memorate mai multe versiuni X1, X2,……Xk ale articolului X, fiecare versiune cu mărcile de citire si de scriere , read_TS(Xi), write_TS(Xi), trebuie să respecte :

1. daca tranzactia T lanseaza o operaţie de scriere a articoluli x şi daca versiunea a lui X are cea mai mare marca de scriere , având TS(<read_TS(Xi), atunci se anuleaza şi ruleaza înapoi tranzacţia T, astfel, se creeaza o noua versiune Xj a lui x cu mărcile read_TS(Xj)=write_TS(Xi)
2. daca tranzactia T lanseaza o operatie de citire a lui x, atunci valoarea returnata este versiunea i a lui X

C.3 Controlul optimist al concurenţei[3]

In tehnica de control optimist al concurentei nu se face nici un test în cursul executiei tranzactiilor, dar actualizarile nu se efectueaza în tabele, ci şi in copii locale ale articolelor

Fazele de execuţie a controlului optimist al concurentei se sintetiza astfel :

1. faza de citire, tranzacţia citeste date din tabele bazei de date, dar modificarile se aplică numai unor copii locale ale articolelor
2. faza de validare : se verifica daca au fost respectate conditiile de serilizabilitate
3. faza de scriere : daca au fost respectate conditiile de serilizabilitate , atunci modificarile efectuate sun aplicate articolelor din tabele BD.

Exemplu :Program tranzactii.sql

```
/*
*         Program PL/SQL pentru  exemplificarea unei tranzacţii
*
*/
CREATE OR REPLACE PROCEDURE  test (val IN number)
AS
-- SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN
  INSERT INTO EDITURI(IdEditura,locsediu,adresa,telefon,mobil,email,wwww)
VALUES(PK_EDITURI.NEXTVAL, 'BUCURESTI', 'STRADA
..','0211234567','0724123456','editura@all.ro','All.ro');
  IF val = 1
    THEN COMMIT;
    ELSE ROLLBACK;
  END IF;

 END;
```

9.2.Crearea schemei de baze de date pe serverul Oracle

În continuare vom transforma arhitectura aplicaţiei pe un server de baze de date Oracle, cu menţiunea că formularele şi rapoartele rămân în Visial Foxpro, instalate pe staţiile reţelei

Procesul de migrare presupune parcurgerea în general a următoarelor etape:

Etapa I Portarea bazei de date în mediul serverului de baze de date care ar trebui realizată în doi paşi:

P1**. crearea schemei bazei de date**, specificarea structurii tabelor, declararea restricţiilor, respectiv cele refrentiale

P2. incarcarea datelor din mediul VFP in baza de date Oracle

Etapa II Configurarea mediului client/server sau, altfel spus, crearea canalului de legatură intre clientul VFP si serverul Oracle

Etapa III Crearea mecanismului de acces si actualizare (regăsire şi modificare/inserare/stergere) a inregistrarilor din baza de date Oracle prin structurile de date specifice clientului VFP

Etapa IV Redefinirea formularelor si rapoartelor ce vor rula pe clientul VFP, astfel incăt să gestioneze corect datele rezidente pe serverul Oracle pe care le vor accesa prin mecanismele din etapa anterioara, respectiv tratarea erorilor

Definitie:Un sistem informatic client-server este un model de lucru in care mai multe programe autonome comunica prin schimb de mesaje

In general, clientii sunt calculatoare personale utilizate pentru activitaţi de gestionare a datelor.Dupa *Sinha* ,un post client se caracterizeaza prin faptul ca :a)reprezinta o interfaţa utilizator care e de obicei grafica (GUI), b)formuleaza

interogari, c)transmite interogarile/comenzile respective serverului prin intermediul unei tehnologii de comunicaţie, d)analizeaza datele din rezultatele de la c)primite de la server, *iar un server prin faptul ca:* a)furnizeaza un serviciu clientului; b)răspunde la interogarile/comenzile clientului; c)ascunde detaliile sistemului client/server, făcând transparent dialogul dintre client şi server

1.Migrarea bazei de date din VFP in Oracle s-ar putea realiza folosind meniul *Tools →wizard →upsizing*, prin selectarea din dialogul wizard *selection a optiuni Oracle upsizing wizard.*
Aceasta ar trebui să refacă baza de date pe o platforma Oracle, redefinând structura tabelelor

Listing 9.2 scriptul de creare a bazei de date biblioteca in VFP
****

Setup.prg
****

```
close data all
close database all
close table all
close all
set default to c:\biblio\database
***se sterge baza de date (dictionarul de date
)biblioteca=biblio
delete database biblio DELETETABLES
**se recreaza baza de date biblio
create database biblio
*se creaza tabelele bazei de date biblio
*tabela  de autori
create table autori(;
grupa char(9);
primary key,;
check(grupa=ltrim(proper(grupa)));
error 'Prima litera trebuie sa fie majuscula,restul
litere mici !',;
cifra number(2);
)
*tabela  clasificare universala zecimala(czu)
create table czu(;
dendiviz char(35);
primary key,;
check(dendiviz=ltrim(proper(dendiviz)));
error'Prima litera din cuvintul denumire diviziune este
majuscula !',;
clasa char(15);
)
*tabela comenzi de carti
create table comenzi(;
unitatea char(25),;
autor char(25),;
titlu char(25),;
precom char(20),;
repartiz char(20),;
mentiuni char(20),;
datacom date,;
```

```
nrex number(4);
)
*tabela  REVISTE -EVIDENTA PRELIMINARA
CREATE TABLE reviste(;
nrinv number(6);
primary key;
check(nrinv>0);
error 'Nr.inventar trebuie sa fie mai mare decat zeo
!',;
titlu char(25),;
check(substr(titlu,1,1))=upper(substr(titlu,1,1));
error 'Prima litera din titlu trebuie sa fie majuscula
!',;
editor char(25),;
loc char(15),;
editura char(15),;
perioada number(4),;
primit char(10),;
cost number(10),;
nrfact number(10),;
nrpag number(4),;
nrex number(4),;
anedi number(4),;
ancal number(4),;
luna char(15);
check(substr(luna,1,1))=upper(substr(luna,1,1));
error 'Prima litera din luna trebuie sa fie majuscula
!',;
index1 char(10),;
mentiuni char(15),;
cota char(20),;
semna char(15),;
issn char(9);
)
*tabela  ZIARE -EVIDENTA PRELIMINARA
CREATE TABLE ziare(;
nrinv number(6);
primary key;
check(nrinv>0);
error 'Nr.inventar trebuie sa fie mai mare decat zeo
!',;
titlu char(25),;
check(substr(titlu,1,1))=upper(substr(titlu,1,1));
error 'Prima litera din titlu trebuie sa fie majuscula
!',;
editor char(25),;
loc char(15),;
editura char(15),;
perioada number(4),;
primit char(10),;
cost number(10),;
nrfact number(10),;
nrpag number(4),;
```

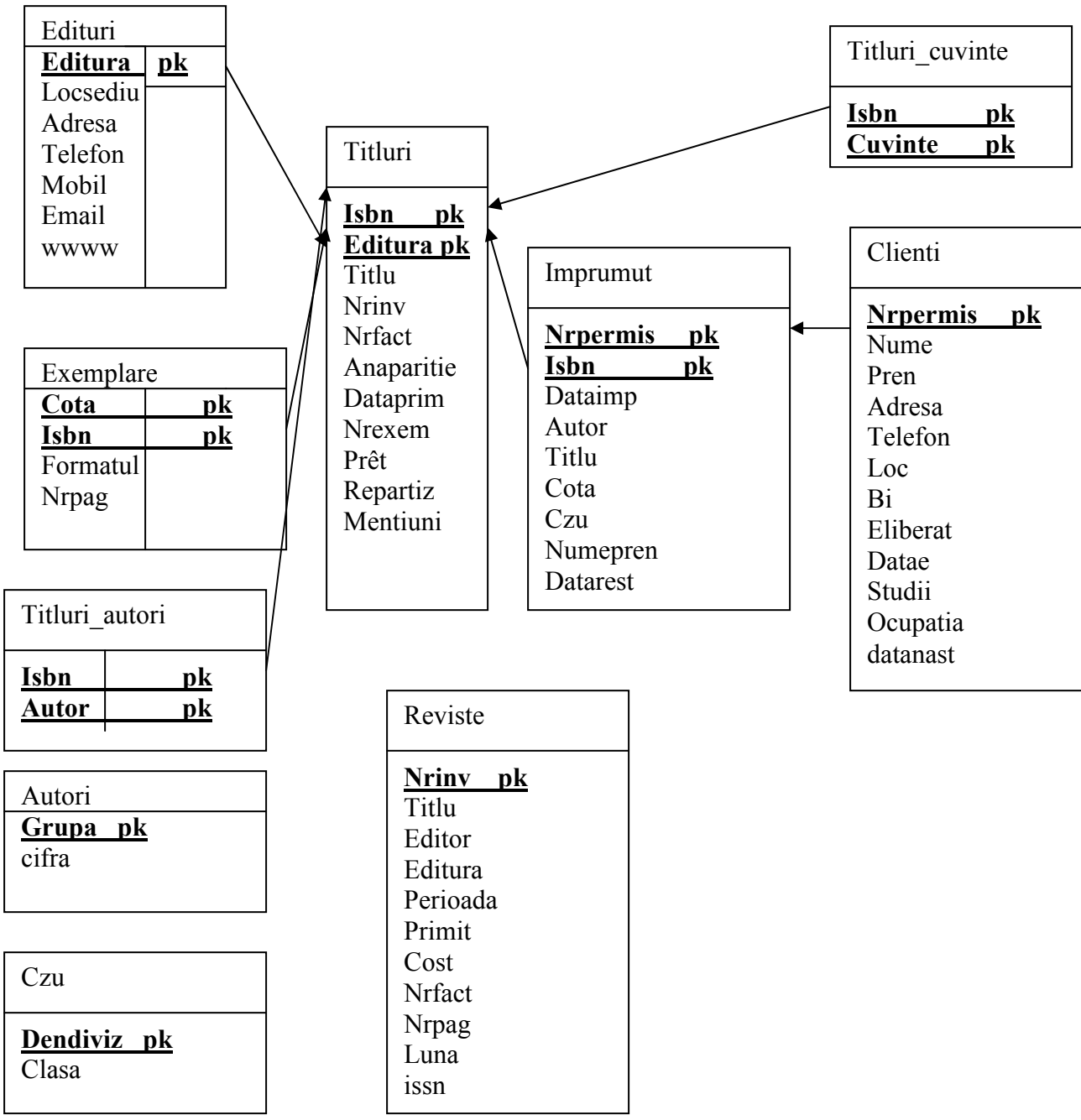
```
nrex number(4),;
anedi number(4),;
ancal number(4),;
luna char(15);
check(substr(luna,1,1))=upper(substr(luna,1,1));
error 'Prima litera din luna trebuie sa fie majuscula
!',;
index1 char(10),;
nrziar number(6),;
dataz date,;
mentiuni char(15),;
cota char(20),;
semna char(15),;
issn char(9);
)

*tabela CARTI EVIDENTA CARTILOR INTRATE- IESITE DIN
BIBLIOTECA
CREATE table carti(;
nrinv number(6);
primary key,;
nrfact number(15),;
dataprim date,;
primirea char(15),;
furnizor char(20),;
adresa char(20),;
www char(15),;
email char(20),;
telefon char(15),;
nrex number(6),;
canti number(6),;
pret number(15,2),;
valoare number(15,2),;
nrpag number(6),;
isbn  char(25),;
datac date,;
mentiuni char(20),;
repartiz char(20),;
czu1 char(20),;
czu2 char(20),;
czu3 char(20),;
czu4 char(20),;
nume1 char(15),;
check(nume1=ltrim(proper(nume1)));
error 'Prima lietra este obligatoriu majuscula !',;
pren1 char(20),;
nume2 char(15),;
pren2 char(20),;
nume3 char(15),;
pren3 char(20),;
autori char(35),;
titlu char(35);
check(substr(titlu,1,1)=upper(substr(titlu,1,1)));
```

```
error 'Prima litera trebuie sa fie majuscula !',;
locul char(15),;
editura char(20),;
anul number(4),;
inaltime numer(5),;
cuvintech char(15),;
cota char(20),;
semna char(20),;
formate char(15),;
cante number(6),;
vale number(15,2),;
cantec number(6),;
valec number(15,2),;
canter number(6),;
valer number(15,2),;
canteu number(6),;
valeu number(15,2),;
cantea number(6),;
valea number(15,2),;
stoc number(15),;
sold number(15,2);
)


*tabela CLIENTI -inscrisi
create table clienti(;
nrpermis number(6) NOT null;
PRIMARY KEY ,;
nume char(25),;
pren char(15),;
adresa char(30),;
telefon char(15),;
loc char(15),;
telem char(15),;
bi char(15),;
eliberat char(15),;
datae date,;
datanast date,;
studii char(15),;
ocupatia char(25);
)


*tabela  edituri
CREATE TABLE edituri(;
editura char(30);
PRIMARY key,;
locsediu char(30),;
adresa char(30),;
telefon char(10),;
mobil char(10),;
email char(30),;
wwww char(30);
```

```
)
*tabela titluri
CREATE TABLE titluri(;
idisbn number(13) not null;
PRIMARY KEY,;
isbn char(13)not null,;
titlu char(60)not null,;
nrinv number(6),;
nrfact number(15),;
anaparitie number(4),;
pret number(12,2),;
dataprim date,;
repartiz char(20),;
mentiuni char(20),;
nrexem number(6),;
editura char(30) NOT null,;
FOREIGN KEY editura TAG editura REFERENCES edituri TAG
editura;
)
*tabela exemplare
CREATE TABLE exemplare(;
idcota number(15) NOT null;
PRIMARY key,;
cota char(15) NOT null,;
formatul char(15),;
nrpag number(4),;
isbn char(13)not null,;
idisbn number(13)not null,;
FOREIGN KEY idisbn TAG idisbn references titluri TAG
idisbn;
)

*tabela titlui_autori
CREATE TABLE titluri_autori(;
idisbn number(13)not null,;
isbn char(13) NOT null,;
autor char(30)not null,;
PRIMARY KEY STR(idisbn,13)+autor TAG primaru,;
foreign KEY idisbn TAG idisbn references titluri TAG
idisbn;
)

*tabela titluri_cuvinte
CREATE TABLE titluri_cuvinte(;
idisbn number(13)not null,;
isbn char(13) NOT null,;
cuvinte char(30),;
primary key STR(idisbn,13)+cuvinte tag primaru,;
foreign key idisbn tag idisbn references titluri tag
idisbn;
)

 *tabela imprumut
```

```
 CREATE TABLE imprumut(;
 nrpermis number(6) NOT null,;
 idisbn number(13) NOT null,;
 isbn char(13) NOT null,;
 nrinv number(6),;
 dataimp date,;
 autor char(30),;
 titlu char(60),;
 czu char(20),;
 cota char(15),;
 numepren char(30),;
 datarest date,;
 PRIMARY KEY STR(nrpermis,6)+STR(idisbn,13) TAG
primaru,;
 FOREIGN KEY nrpermis TAG nrpermis REFERENCES clienti
TAG nrpermis;
 )

*tabela REGISTRUL DE INVENTAR AL CARTILOR DIN
BIBLIOTECA
create table inventar(;
nrinv number(6);
primary key,;
datai date,;
autor1 char(35),;
autor2 char(35),;
autor3 char(35),;
autorul char(35),;
titlul char(40),;
locul char(20),;
editia char(15),;
editura char(25),;
anul number(4),;
pret number(15,2),;
czu char(15),;
semn char(12),;
cota char(30),;
nrcrti number(10),;
nrcrtii number(10),;
mentiuni char(25);
);
```

Pasul 1 Schema bazei de date BIBILOTECA
1)Prezentarea schemei bazei de date BIBLIOTECA



1)Tabela Autori –conţine informaţii despre nomenclatorul de autori –numită tabela de autori a lui Cutter Ch .Numarul din tabela, corespunzator primei sau primelor silabe (grupe de litere) a numelui de autor sau titlului puclicatiei, se

adauga la initiala autorului sau primului cuvant din titlului, se obţine semnul de autor

Tabela de autori are urmatoarea structura :

- Grupa,c,9 –reprezinta grupa de autori, fiind si cheia primara
- Cifra,n,2 –cifra corespunzatoare tabelei

Exemplu

Grupa	Cifra	Grupa
A	10	B
Aa	11	Ba
Abe	12	Bac

Exemple de codificare semn autor

2)Tabela CZU (Clasificarea zecimala universala)-contine totalitatea cunostintelor umane ca o singura unitate imparţita in zece mari clase, notate cu cifra arabe, de la 0-9, iar fiecare clasa se imparte in 10 subdiviziuni etc…..

Cota se noteaza pe fisa cu o linie orizontala desparţitoare la mijloc şi se completeaza in colţul stanga-sus, iar in colţul stanga –jos indicele CZU, care indica locul fisei in catalogul systematic

Tabela CZU are urmatoarea structura :

-dendiviz,c,35-subdiviziunea carti-fiind cheia primara, fără spatii la inceput de text

-clasa,c,15-reprezinta clasa

Exemplu

Dendiviz	Clasa
Generalitati	0
Informatica	51
Comert.relatii economice internationale.	339

3)Tabela EDITURI-reprezinta nomenclatorul editurilor de publicaţii având urmatoarea structura de baza

- Editura,c,30-reprezinta editura (de exemplu:ALL,POLIROM,TEHNICA ETC….), care reprezinta si cheia primara
- Locsediu,c,30-locul sediu al editurii (ex:Bucuresti,Iasi….)
- Adresa,c,30-adresa completa a editurii (strada,bloc,nr,ap etc….)
- Telefon,c,10-reprezinta telefonul fix
- Mobil,c,10-telefonul mobil;
- Email,c,30-adresa de e-mail al editurii
- Wwww,c,30-adresa de internet al editurii

Editura	Locsediu	Adresa	Telefon	Mobil	Email	Www
All	Bucuresti	b-dul Timisoara	0214130715	0724123456	edit@all.ro	All.ro

4)Tabela Comenzi contine informatii despre comenzile facute de catre o biblioteca,avand structura:

- Editura,c,30-reprezinta editura de publicatie
- Autor,c,35 –autorul cartii
- Titlul,c,60 –titlul publicatiei
- Repartiz,c,20-reprezinta repartizarea cartilor conform structurii,anume:CZU1/3,902/904,908,93/99-Filosofie,czu 5/6,91-stiinte

exacte, tehnica, geografie, czu 8-lingvistica, czu 0,7,929-generalitati etc….

- Mentiuni,c,30-mentiunea cartii
- Datacom,date-data comenzii
- Nrex,n,4-numarul de exemplare oferite/cerute

5)Tabela TITLURI –contine informatii despre cartile intrate/iesite /interschimbate intr-o biblioteca, având urmatoarea structura de baza:

- ISBN,C,13-NOT NULL-reprezinta numarul standard international pentru carti, fiind şi cheia primara de exemplu (ISBN=973-9392-46-6ETC…), iar pentru reviste, ziare periodice se utilizeaza ISSN
- TITLUL,C,60-titlul cartii (de exemplu:sistem de gestiune a bazelor de date –aplicatii)
- Nrinv,n,6-reprezinta numarul de inventar scris pe carte
- Nrfact,n,15-reprezinta numarul documentului de intrare (factura,aviz,etc….)
- Anaparitie,n,4-anul aparitiei ☹ex:2005,2006)
- Prêt,n,12,2-pretul cartii
- Dataprim,date-data primirii cartii in biblioteca sub forma zz.ll.aaaa
- Repartiz,c,20-repartizarea dupa continut ,vezi tabela de comenzi
- Mentiuni,c,20-la fel;
- Nrexem,n,6-reprezinta numarul de exemplare intrate/iesite/interschimbate intr-o biblioteca
- Editura,c,30-editura care reprezinta o cheie straina pentru tabela edituri

6)Tabela Exemplare-contine date despre cota topografica ,respectiv formatul care are urmatoarea structura:

- Cota,c,15,not null –cotoa topografica prezentata in tabela clasificarea zecimala univcersala(CZU), fiind si cheia primara
- Formatul c,15-formatul (inaltimea publicatiei)-asezara cartilor in depozite pe rafturi,avand urmatoarele informatii :
  -Formatul I-<20cm
  -Formatul II –de la 20 cm-la 25 cm
  -Formatul III-de la 25-la 30 cm
  -Formatul IV –de la 30-38 cm
  -Formatul V peste 38 cm

Marcat in cota prin cifre romane de exemplu I 3214, V 324-formeaza cota topografica, pentru diferite categorii de tipărituri se utilizeaza alte litere P(periodice),M(muzica),H(harta),Ars(arta)etc…..

- Nrpag,n,4-inseamna numarul de pagini
- Isbn,c,13 numarul standard international penru carti-fiind cheia starina

7)Tabela titluri_autori –contine informatii despre autorii publicaţiei ,având structura:

- Isbn,c,13-not null –cheie straina decalarata prin references titluri tag isbn;
- Autor,c,35-reprezinta autorul cartii –not null cu mentiunea ca prima litera trebuie sa fie masjuscula si fara spatii la inceput
- Cheia primara fiind isbn+autor

8)Tabela titluri_cuvinte-contine informatii despre cuvintele cheie ale cartilor,anume:

(bazae de date relationale,aplicatii distribuite realizate cu Java…),cu structura:

- Isbn,c,13
- Cuvinte,c,30-reprezinta cuvintele cheie ale publicatiei
- Cheia primara=isbn+cuvinte

9)Tabela Clienti-reprezinta tabela cu informatii despre inscrierea cititorilor intr-o biblioteca,cu urmatoarea structura:

- Nrpermis,n,6-numar permis de intrare care este unic,fiind si cheia primara a tabelei
- Nume,c,25;pren,c,15-numele /prenumele cititorului (elevului din cadrul scolii/profesorului)
- Adresa,c,30-adresa completa conform actului de identitate
- Telefon,c,10-numar de telefon fix,mobil,acasa,scoala,serviciu;
- Loc,c,15-locatitata
- Bi,c15-seria buletinului de identitate si seria
- Eliberat,c,15-date,date-data eliberarii
- Studii,c,15-studii {elev,student,……}
- Ocupatia,c,25-ocupatia de baza a profesorului,elevului etc…

10)Tabela imprumut de carti –reprezinta fişa centrala de contact de imprumut de carţi catre cititori inregistrati,avand structura de baza:

- Nrpermis,n,6-fiind cheia primara de la tabela clienti,daca nu exista in baza de date atunci nu se poate imprumuta o carte
- Nrinv,n,6-numar inventar inregistrat pe carte
- Dataimp,date-data imprumutului sub forma zz/ll/aaaa
- Autor,c,35-autorul cartii de imprumutat
- Titlu,c,60-titlul cartii de imprumutat
- Czu,c,20-indicele de cota prezentat in tabela CZU
- Cota,c,15-cota topografica prezentata anterior
- Numepren-numele /prenumele cititorului adaugat autmat din tabela clienti
- Datarest, date (zz/ll/aaaa)-reprezinta data restiturii cartii, fiind atributul cel mai important ,care o conditie de 15 zile –necesara in stabilirea listei restantelor de carti , conform fisei cartilor
- Cheia prima este formata din str(nrpermis,6)+str(nrinv,6)

11)tabela de vederi :fisa cartii,fisa de imprumut,lista restantelor,reviste,ziare
Fisa de catalog are dimensiuni de stas international,iar in lipsa lor se folosesc listele obisbuite listate la imprimanta

Fisa de catalog

Descrierea bibliografica prezinta elementele esentiale de identificare a publicatiei:titlul, autorul/autorii sau responsabili de lucrare,editia, date de publicare (loc,editura,an), descrierea cantitativa(paginatia), colectia, alte note si numarul standard (ISBN) pentru carti si ISSN pentru periodice
In afara de aceste elemente de descriere bibliografica, pe fiecare fisa se stabileste cate o vedeta,vedeta este cuvintul care da ordinea de intrare a fisei in catalog si care se scrie pe primul rand de la prima verticala
In vedeta,numele autorilor personali se scriu mai intai cu numele si apoi cu prenumele
Schema generala este:

VEDETA
Titlul:informatii la titlu/mentiune de responsabilitate(autorii asa cum sunt trecuti pe carte),-editia.
-loc publicare:editura,anul publicarii.
Paginatie:ilustratii.-(colectia;numarul in cadrul colectiei).
Note.
I.S.B.N
Punctuatia din schema de mai sus trebuie respectat intocmai:
Exemplu:
Programare avansata in Oracle 9i/Ileana Popescu,Alexandra Alecu,Letita Velescu,Gabariela Florea
Bucuresti:Editura Tehnica,2004
Bibliografie.
ISBN 973-31-2208-4
I..Popescu,Ileana
II..Alecu,Alexandra
III.Velescu,Letita
IV.Florea,Gabriela

004.42 ORACLE9i

12)Oraganizarea cataloagelor
In bibliotecile mici sunt folosite doua tipuri de *cataloage:*
a)*catalogul alphabetic*-organizat dupa numele autorilor sau titlurilor de anonime;
b)*catalogul systematic*-structurat dupa clasificarea zecimala universala
Convertirea programului scris în VFP în ORACLE
Vom converti acest script tinând cont de urmatoarele caracteristici ale limbajului *SQL-DDL* propriu *Oracle:*

- Frazele SQL din Oracle se pot intinde pe mai multe randuri (ndespartiţe prin caractere speciale cum este ; in VFP), finalul lor fiind declarat prin caracterul ";"
- In fraza *CREATE TABLE* din VFP, restricţiile de la nivelul cămpurilor sunt despartite prin caracterul ",", iar cămpurile –prin lipsa oricarui character special în mediul Oracle insa,restricţiile nu sunt despartite prin ",", prezenţa acestuia semnificănd incheierea declaraţiilor pentru un cămp (adica nume+tip data+restricţii) sau incheierea declaraţiilor pentru fiecare restricţie de la nivelul intregii tabele
- In ceea ce priveste tipurile de date, vom respecta urmatoarele reguli:
  -tipul VFP *number* va corespunde cu *number d*in Oracle
  -tipul char va corespunde in Oracle daca respectivul cămp face parte dintr-o cheie primara sau straina,sau cu varchar2
  -pentru tipul logic logic nu exista in Oracle ,atunci acesta poate fi convertit in char ,fie in number ,iar respectivul cămp va fi declarat un check care are doua valori (.t.=adevarat,f.=false)
  -tipurile *date* şi *datetime* vor fi convertite in DATE al Oracle
- In ceea ce priveste restricţiile din VFP
  -VFP Primary key –Oracle PRIMARY KEY,VFP Check-Oracle check,vfp NOT NULL-Oracle not null, VFP foreign key– oracle foreign key vfp unique-oracle unique key, funcţii:
- -funcţia PROPER din VFP –INITCAP Oracle
- -functia INLIST() din VFP-va fi transformată in ORACLE printr-o expresie continând operatorul IN;

- -funcţia DATE() din VFP (data sistemului)-Oracle este SYSDATE
  Funcţia between() din VFP –BETWEEN din Oracle;
  - <expresie evaluate> BETWEEN <expresie limita inferioara>AND<expresie limita superioara>;
  - invocarea unei valori tip data calendaristica in VFP prin genul de forma({01/01/2006} va fi tradusa in Oracle prin TO_DATE
  - in restricţia PRIMARY KEY,UNIQUE şi FOREIGN KEY din Oracle sunt specificate cămpurile şi eventual tabele participante sunt expresii de indexare ca in VFP,construirea şi gestionarea expresiilor indecşilor corespunzatori nefiind transparenţa dezvoltatorilor de exemplu expresia din VFP
  - PRIMARY KEY isbn+autor funcţie tag primar1 va corespunde in Oracle cu

CONSTRAINT PRIMARy primary key **(isbn,autor)**

Listing 9.2.2 crearea_bd_biblioteca_oracle.sql script pentru crearea bazei de date Oracle

```
drop table imprumut;
drop table clienti;
drop table titluri_cuvinte;
drop table titluri_autori;
drop table exemplare;
drop table titluri;
drop table edituri;
drop table czu;
drop table autori;
drop table reviste;
drop table ziare;
drop table carti;

create table autori(
grupa varchar2(9) primary key
,cifra number(2)
)
/


create table czu(
dendiviz varchar2(35) primary key
,clasa char(15)
)
/

create table edituri(
editura varchar2(30) primary key
,locsediu varchar2(30)
,adresa  varchar2(30)
,telefon char(10)
,mobil char(10)
,email varchar2(30)
,wwww varchar2(30)
)
```

```
/

create table titluri(
idisbn number(13) not null primary key
,titlu varchar2(60) not null
,editura varchar2(30) not null references edituri(editura)
,nrinv number(6)
,nrfact number(15)
,anaparitie number(4) default extract (year from current_date)
,dataprim date default sysdate
,nrexem number(4)
,pret number(12,2)
,repartiz varchar2(20)
,mentiuni varchar2(20)
)
/

create table exemplare (
idcota number(15) not null primary key
,formatul varchar2(15)
,nrpag number(4)
,idisbn number(13) not null references titluri(idisbn)
)
/

create table titluri_autori(
idisbn number(13) not null references titluri(idisbn)
,autor varchar2(30) not null
,primary key(idisbn,autor)
)
/

create table titluri_cuvinte(
idisbn number(13) not null references titluri(idisbn)
,cuvinte varchar2(30)
,primary key(idisbn,cuvinte)
)
/

create table clienti(
nrpermis number(6) not null primary key
,nume varchar2(25)
,pren varchar2(15)
,adresa varchar2(30)
,telefon char(10)
,loc varchar2(15)
,telem char(10)
,bi char(15)
,eliberat varchar2(15)
,datae date default sysdate
,datanast date default sysdate
,studii varchar2(15)
,ocupatia varchar2(25)
```

```
)
/

create table imprumut(
nrpermis number(6) not null references clienti(nrpermis)
,idisbn number(13)
,nrinv number(6)
,dataimp date default sysdate
,autor varchar2(30)
,titlu varchar2(60)
,czu varchar2(20)
,cota varchar2(15)
,numepren varchar2(30)
,datarest date default sysdate
,primary key(nrpermis,idisbn)
)
/



create table reviste(
nrinv number(6)
,titlu varchar2(60)
,editor varchar2(25)
,loc char(15)
,editura1 varchar2(15)
,perioada number(4)
,primit char(10)
,cost number(10)
,nrfact number(10)
,nrpag number(4)
,nrex number(4)
,anedi number(4)
,ancal number(4)
,luna varchar(15)
,index1 char(10)
,mentiuni varchar2(15)
,cota varchar2(20)
,semna char(15)
,issn char(9)
)
/



create table ziare(
nrinv number(6)
,titlu varchar2(60)
,editor varchar2(25)
,loc char(15)
,editura1 varchar2(15)
,perioada number(4)
,primit char(10)
,cost number(10)
```

```
,nrfact number(10)
,nrpag number(4)
,nrex number(4)
,anedi number(4)
,ancal number(4)
,luna varchar2(15)
,index1 char(10)
,mentiuni varchar2(15)
,cota varchar2(20)
,semna char(15)
,nrziar number(6)
,issn char(9)
)
/

drop table inventar;

create table inventar(
nrinv number(6) not null primary key,
datai date default sysdate,
autor1 varchar2(35),
autor2 varchar2(35),
autor3 varchar2(35),
autorul varchar2(35),
titlul varchar2(60),
locul varchar2(20),
editia varchar2(15),
editura varchar2(30),
anul number(4),
tehnica varchar2(20),
formatul varchar2(20),
pret number(15,2),
cota varchar2(15),
czu varchar2(15),
semn varchar2(15),
nrcrti number(10),
nrcrtii number(10),
mentiuni varchar2(25)
)
/
```

Listing 9.2.3 generare_script_bd_oracle.prg
Program pentru popularea bazei de date oracle

```
PROCEDURE generare_script_bd_oracle
PARAMETERS numebazadate,dirscriptgenerat
LOCAL
nrtable,a_table(1),nume_script_pop_tbl,numescriptgenera
t
IF !dbUSED(numebazadate)
OPEN DATABASE (numebazadate) SHARED
```

```
ENDIF
IF EMPTY(dirscriptgenerat)
dirscriptgenerat=SYS(5)+SYS(2003)
ENDIF
numescriptgenerat=dirscriptgenerat+'\populare_bd_'+nume
bazadate+'.sql'
MESSAGEBOX('atentie ! veti obtine un script localizat
in '+numescripgenerat+'!')


nrtable=ADBOBJECTS(a_table,"table")

SET TEXTMERGE on
SET TEXTMERGE TO (numescriptgenerat) noshow
\\<<DATETIME()>>

FOR i=1 TO ALEN(a_table,1)
IF EMPTY(numescriptgenerat)

nume_script_pop_tbl=SYS(5)+SYS(2003)+'\populare_'+a_tab
le(i)+'.sql'
ELSE
nume_script_pop_tbl=dirscriptgenerat+'\populare_'+a_tab
le(i)+'.sql'
ENDIF
\@@
\\<<nume_script_pop_tbl>>
ENDFOR
\
\commit;
set textmerge to
SET TEXTMERGE off

FOR j=1 TO ALEN(a_table,1)
generare_insert_sql(a_table(j),dirscriptgenerat)
ENDFOR
ENDPROC

PROCEDURE generare_insert_sql
PARAMETERS numetabela,dirscript
LOCAL
nrcampuri,a_campuri(1),sir_clauza_campuri,sir_clauza_va
lues,sir_sql_cmd
CLEAR
SET STRICTDATE to 0
SET DATE TO britsh
SET CENTURY on
IF !USED(numetabela)
USE (numetabela) IN 0 SHARED
ENDIF
nrcampuri=AFIELDS(a_campuri.numetabela)
SELECT (numetabela)
sir_clauza_campuri=''
```

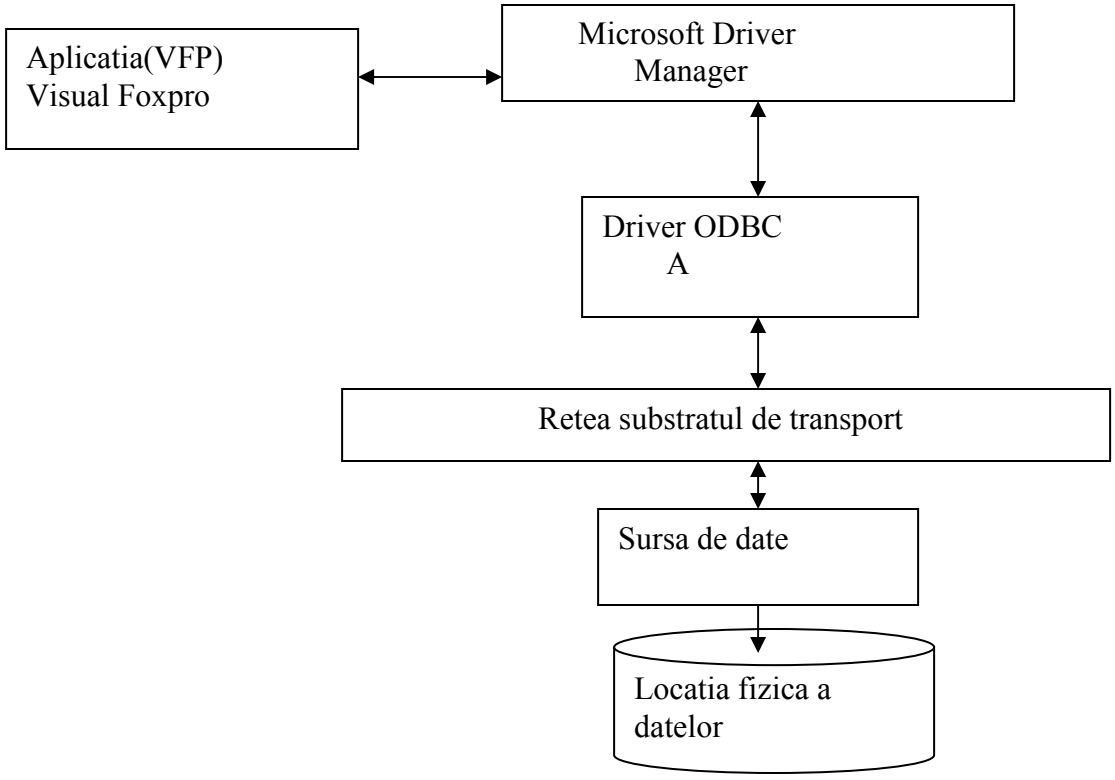
```
sir_clauza_values=''
sqlcmd=''
IF EMPTY(dirscript)
numescript=SYS(5)+SYS(2003)+'\populare_'+numetabela+'.s
ql'
ELSE
numescript=dirscript+'\populare_'+numetabela+'.sql'
ENDIF
?'atentie !veti obtine un script localizat in
'+numescript+'!'


FOR i=1 TO ALEN(a_campuri,1)
   IF INLIST(a_campuri(i,2),'N','C','D','T','L')

sir_clauza_campuri=sir_clauza_campuri+a_campuri(i,2)
      IF i<> ALEN(a_campuri,1)
      sir_clauza_campuri=sir_clauza_campuri+","
      ENDIF
      ENDIF
      ENDFOR

      SET TEXTMERGE on
      SET TEXTMERGE TO (numescript) noshow
      \\<<DATETIME()>>
      nrinreg=RECCOUNT()
      SCAN


      FOR i=1 TO ALEN(a_campuri,1)
   IF INLIST(a_campuri(i,2),'N','C','D','T','L')

sir_clauza_campuri=sir_clauza_campuri+a_campuri(i,2)
      IF i<> ALEN(a_campuri,1)
      sir_clauza_values=sir_clauza_values+","
      ENDIF
      ENDIF
      ENDFOR
      sqlcmd='insert into
'+numetabela+'values('+sir_clauza_values+');'
      WAIT WINDOW
'asteptati<<<'+ALLTRIM(STR(RECNO()))+'/'+ALLTRIM(STR(nr
inreg)) nowait
      \<<sqlcmd>>
      sir_clauza_values=''
      ENDSCAN
      \
      \COMMIT;
      SET TEXTMERGE TO
      SET TEXTMERGE OFF
      WAIT WINDOW "GATA !" NOWAIT
      retu
```

```
      FUNCTION FORMAT_expresie_valoare
      PARAMETERS numevar
      DO case
      CASE ISNULL(&numevar)
      RETURN 'null'
      CASE VARTYPE(&numevar)='N'
        RETURN ALLTRIM(STR(&NUMEVAR,16,2))
        CASE VARTYPE(&NUMEVAR)='C'
        RETURN CHR(39)+ALLTRIM(&NUMEVAR)+CHR(39)
        CASE VARTYPE(&NUMEVAR)='D' OR
VARTYPE(&NUMEVAR)='T'
        RETURN

'TO_DATE('CHR(39)+DTOC(&NUMEVAR)+CHR(39)+','+CHR(39)+'D
D/MM/YYYYHH:MI:SS'+CHR(39)+')'
        CASE VARTYPE(&NUMEVAR)='L'
        RETURN ALLTRIM(STR(IIF(&NUMEVAR,1,0)))
        OTHERWISE
        RETURN '<unknown>'
        ENDCASE
   Return
```

9.3.Configurarea mediului client-server Client-VisualFoxpro(VFP)-server Oracle 9i

9.3.1.ODBC-drivere ODBC [7]

OpenDatabase Conncevtivy(ODBC), reprezintă o intefaţa care permite unei aplicaţii să acceseze date din surse diferite: *Oracle ,MS SQL server,….*
Legatura intre aplicatiile ce exploateaza aceste surse de date si serverele BD se realizeaza prin drivere specifice, care reprezinta de fapt ,DDL-uri invocate atunci cand se solicita accesul la o anumita sursa de date, deci prin urmare se poate conecta la oricare sursă pentru care există drivere.
Figura componentele ODBC



Interfata ODBC defineste :

1. o biblioteca de apeluri de funcţii care permit unei aplicaţii să se conecteze la o sursa de date, se execute fraze SQL şi să primeasca rezultatele acestora ;
2. modalitate standard de conectare la o sursa de date externa ;
3. modalitate standard de reprezentare a tipurilor de date
4. receptionarea printr-un mecanism standardizat al erorilor generate de motoarele diferitelor servere de baze de date care gestioneaza BD sursa
5. traducerea specificatiilor SQL in specificatii proprii serverelor ce gestioneaza datele apartine tot driverelor ODBC

9.3.2.Clientul ORACLE şi protocolul de conectare NET [7]

Într-o configuraţie client-server, dialogul dintre cele doua entităţi pot fi rezumat astfel :

Când un client trimite o cerere (fraza SQL) catre un server BD, aceasta o receptioneaza, o execută, dupa care trimite ca raspuns rezultatul frazei SQL sau erorile care rezulta in urma execuţiei.

O astfel de comunicare simpla şi comodă intre clienti şi serverele de baze de date este posibila in tehnologia Oracle prin protocolul NET.

Protocolul NET se sprijina pe substratul de transport din stiva protocoalelor de reţea (TCP/IP,SPX/IPX) furnizeaza in principal trei functii de baza care se refra la :

1. operaţiile de conectare
2. operaţiile privitoare la date (transferul datelor intre client si server)

3. operaţiile legate de exceptiile evenimentelor anormale ce pot surveni în cadrul unei sesiuni client-server

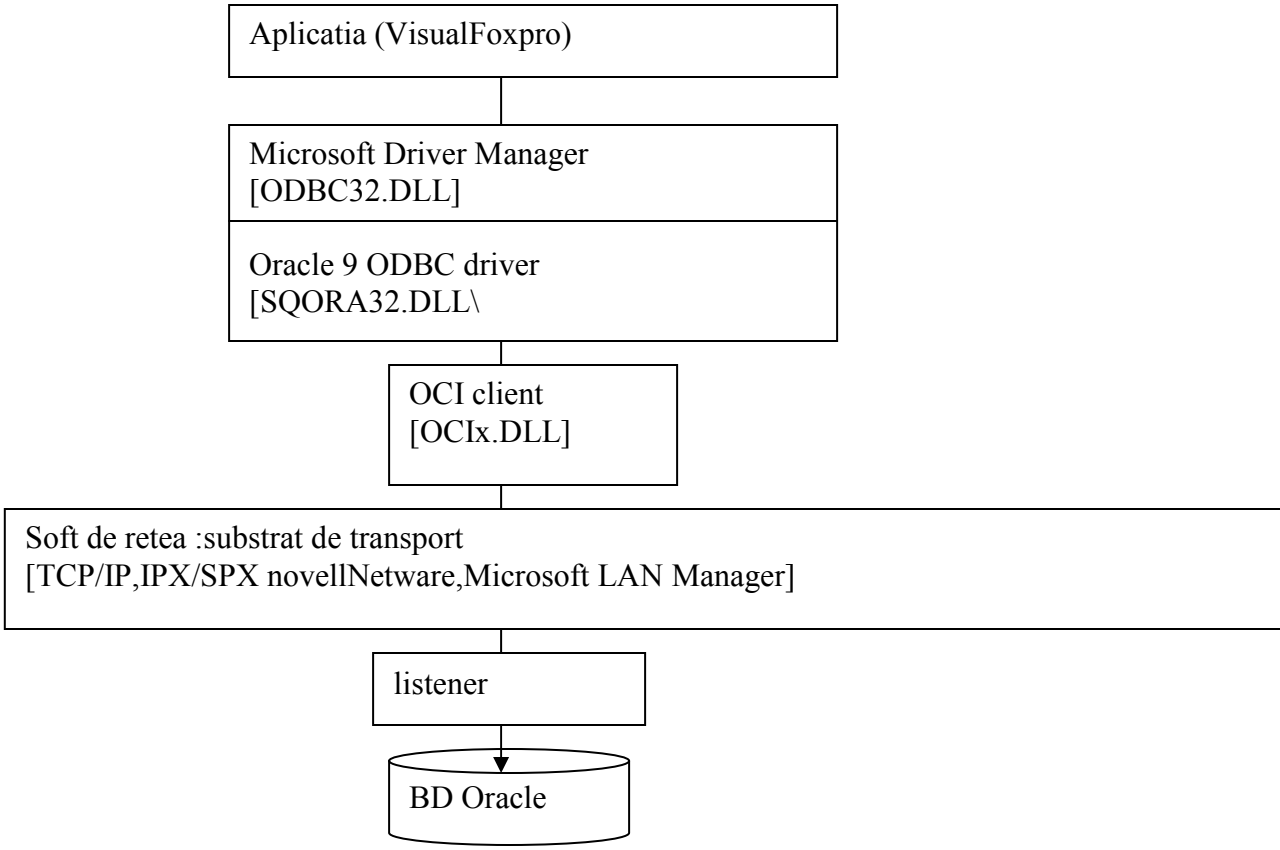
O sesiune de lucru Oracle nu poate fi descrisa fara a avea o imagine, a ceea ce se intampla ce cererile clientilor o data ajunse pe masina care ruleaza serverul Oracle 9i.
Rolul esential în aceasta privinta il are un serviciu special în (WINDOWS 2000,2003) numit **LISTENER**
Aceasta receptioneaza cererile trimise către serverul Oracle prin diferitele protocoale de retea şi le redirecţioneaza (ruteaza) catre bazele de date carora le sunt adresate efectiv.
Procedura de conectare a unui client la un server Oracle se executa in urmatorii pasi :

1. un program sau aplicaţie-utilizator initiaza o cerere de conectare trimitând numele de utilizator, parola şi un nume de serviciu(service name), care poate fi asimilat unui alias al bazei de date destinatie, acest service name este mapat direct pe un descriptor de conectare (*connect descriptor*), care este in sarcina protocolului NET ,care va incearca sa o rezolve citind fişierele de configurare in care se gasesc specificatiile descriptorilor de conectare si metoda de rezolutie
2. dupa ce semnificatia numelului de serviciu este determinata , cererea de conectare este transmisă , prin protocolul de retea existent, serviciului LISTENER de calculatorul central server
3. sesiunea de lucru dintre client şi serverul bazei de date nu este insă sustinuta de catre LISTENER, rolul acestuia limitandu-se doar la redirectionarea cererilor catre instantele bazelor de date carora le sunt adresate, astfel, cand primeste o cerere catre o BD LISTENER –ul creaza un proces server căruia i-o predă controlul sesiunii deschise intre server si client
4. adresa procesului nou-creat este trimisă procesului de maşina clientului, astfel cele doua procese procesul –utilizator care a intiat cererea de conectare – şi procesul server vor comunica direct pe durata sesiunii fara a implica serviciul LISTENER

9.3.3.Arhitectura funcţionala a driverului Oracle ODBC driver[7]

Oracle ODBC Driver prermite aplicatiilor Windows (2000,2003) să efectueze operatii de scriere (actualizare) sau citire (introgare) in BD Oracle folosind softul de comunicare proprietar Oracle, NET

Acest driver foloseste interfata OCI(Oracle Call Interface) de pe client pentru a trimite cererile de acces ale aplicatiei şi pentru a receptiona rezultatele acestora de la sursa de date invocate, acest protocol este utilizat pentru realizarea legaturilor intre clientul OCI si serverul Oracle.
Arhitectura functionala a mecanismului de comunicare prin Oracle ODBC driver este
Figura mecanismul de comunicare prin Oracle ODBC driver



OCI (adica functiile din biblioteca oci.dll) contine toate informatiile pentru initierea si desfasurarea dialogului client-server ORACLE care defineste apeluri catre server pentru :

- Descrierea continutului cămpurilor returnate pe baza informatiilor din dictionarul de date,pentru analiza(pase) frazelor SQL din punct de vedere sintactic ;
- Deschiderea unui cursor ;
- Executarea frazelor SQL in spatiul de memorie destinat cursorului creat ;
- Aducerea(fetch) unei inregistrari sau a mai multor inregistrari in aplicatia client
- Inchiderea cursorului creat
- Aplicatia client utilizeaza o combinatie a acestor apeluri pentru a trimite se executa cereri pe serverul Oracle

1 Crearea unei surse de date ORACLE prin ODBC

Un client ODBC utilizeaza interfata API a driverului ODBC pentru a apela funcţiile specifice acesteia in scopul transmiterii frazelor SQL catre serverele de date

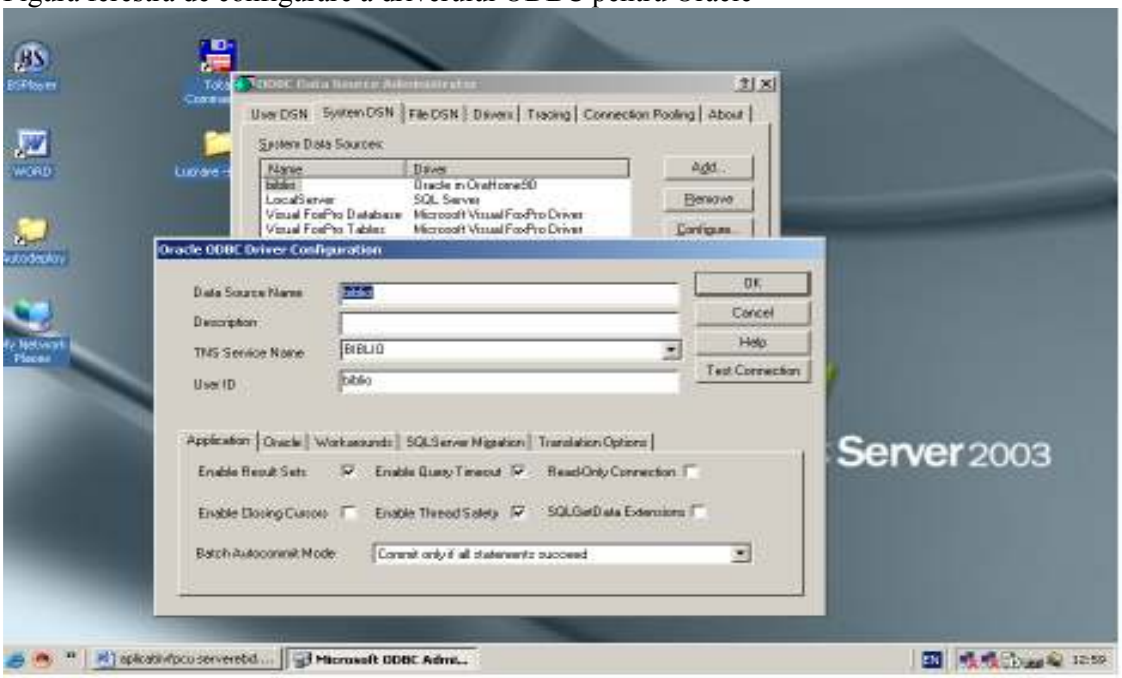
Etapele de configurare sunt :

- trebuie apelat administratorul surselor de date ODBC din control panel-→ODBC data source
- din cadrul de pagina user/system/file se alege butonul add ;
- din lista de drivere ODBC se alege Oracle ODBC driver

Pasul 2 –completarea informatiilor cerute in fereastra de dialog (Oracle ODBC setup)sau (microsoft ODBC for Oracle Setup)

- data sursei de lucru *Biblio, userid si service name (sau connect string ori server* pentru driiverele Microsoft) biblio, biblio--→ok

Figura ferestra de configurare a driverului ODBC pentru Oracle



2 crearea unei conexiuni VisualFoxpro(/clientOracle)

O conexiune (un obiect de tip conncetion) poate fi asimilata definitiei unei surse de date stocate intr-o baza de date VFP.Deschiderea unor astfel de sesiuni se realizeaza fie implicit prin **remote views (tabele derivate la distanta), fie explicit prin interfaţa de transfer SPT(SQL Pass-Throught).**

Definirea unui obiect de tip conncetion intr-o baza de date VFP va cuprinde o serie de parametri care se refera la specificarea sursei de date pentru care este construita conexiunea VFP şi o alta serie de parametri care se refră la gestionarea comunicării/traficului dintre clientul VFP şi sursa de date remote (la distanta).

Crearea si configurarea unei asemenea conexiuni se poate realiza fie asistat, cu ajutorul connection designer-ului vezi figura, fie manual cu ajutorul comenzilor

CREATE CONNCETION SI DBSETPROP

(parametri de configurare al unui obiect de tip connection se stocheaza in dictionarul bazei de date)

Legarea VFP cu Oracle, presupune existenta sau, crearea unei BD şi pe client, pentru stocarea informatiilor privitoare la conexiune şi a definitiilor tabelelor derivate la distanta.

Precizarea sursei de date se poate face prin indicarea explicita a

data source=biblio,userid=biblio,password=biblio,database=biblio
DSN=BIBLIO,UID=BIBLIO,PWD=BIBLIO,DBQ=BIBLIO
Crearea conexiunii prin program :
CREATE CONNECTION BIBLIOTECA ;

DATASOURCE BIBLIO;
USERID BIBLIO PASSWORD BIBLIO DATABASE BIBLIO
*****sau**
CREATE CONNECTION BIBLIOTECA;
CONNSTRING
'DSN=BIBLIO;UID=BIBLIO;PWD=BIBLIO;DBQ=BIBLIO'

Specificaţiile pentru gestionarea comunicării dintre client VFP şi baza /sursa de date Oracle se refera în principal la: *suport pentru tranzactii, procesare sincrona/asincrona,modul de lucru batch/non-batch*
Tranzactiile sunt gestionate fie automat, fie manual cu ajutorul interfeţei SPT prin funcţiile ***SQLCOMMIT() Si SQLROLLBACK()***
Modalitatile de prelucrare a datelor din sursele/bazele de date Oracle, sincrona sau asincrona se referă la modul in care se returneaza controlul aplicatiei dupa apelarea unei functii ce implica trafic ODBC.

Modul de lucri batch sau non-batch stabileste modul de executie al unei funcţii care poate returna mai multe seturi de date dintr-o sursa de date ODBC.
Parametrii privitori la aceste specificatii se pot configura explicit pentru o conexiune in dictionarul bazei de date, cu ajutorul funcţiei DBSETPROP. Prin aceasta functie se pot stabili atributele :

- Transaction –pentru managementul tranzactiilor, ce poate lua valorile 1(DB_TRANSAUTO din foxpro.h) sau 2 (DB_TRANSMANUAL din foxpro.h) ;
- Asynchronous-pentru care valoarea .f. specifica o conexiune sincrona, iar .t. una asincrona
- Batchmode-care setat pe .t. indica o conexiune care lucreaza î modul batch

9.4Actualizarea tabelelor din baza de date gestionata de catre serverul ORACLE
Pentru a actualiza datele, respectiv regasi/interoga datele stocate in tabelele relationale ale serverului Oracle se poate apela la doua mecanisme :
1.tabele derivate la distanta-remote data view ;
2.SQL Pass-Through-care ofera flexibilitatea unui acces mai exact asupra canalului si structurilor de date prin care se face legatura cu serverul bazei de date Oracle
9.4.1.Tabele derivate la distanţa
Crearea tabelei derivate la distanta vedituri
Listing 9.4.1 creare_tabderiv_edituri.prg

```
*crearea tabelei derivate vedituri
#include foxpro.h
CREATE SQL  VIEW vedituri CONNECTION
c:\biblio\database\orabiblio.dsn as select * from
edituri
=DBSETPROP('vedituri','view','tables','edituri')
=DBSETPROP('vedituri.editura','field','keyfield',.T.)
=DBSETPROP('vedituri.editura','field','updatable',.T.)
=DBSETPROP('vedituri.locsediu','field','updatbale',.T.)
=DBSETPROP('vedituri.adresa','field','updatable',.T.)
=DBSETPROP('vedituri.telefon','field','updatbale',.T.)
=DBSETPROP('vedituri.mobil','field','updatbale',.T.)
=DBSETPROP('vedituri.email','field','updatable',.T.)
```

```
=DBSETPROP('vedituri.wwww','field','updatbale',.T.)
=DBSETPROP('vedituri','view','updatetype',DB_UPDATE)
=DBSETPROP('vedituri','view','wheretype',DB_KEYANDMODIF
IED)
=DBSETPROP('vedituri','view','sendupdates',.T.)
```

Parametrul wheretype a fost setat pe valoarea DB_KEYANDMODIFIED, aceasta inseamna ca la comiterea buffer-ului in tabela sursa , se verifica daca intre timp,au fost modificate de alta statie de lucru, atributele(atributul) din cheia primara si atributele modificate local, daca da se declansează eroarea ***Update Conflict***

In general toate proprietatiile tabelei derivate sunt memorate în dictionarul (containerul BD).
Deschiderea se face astfel : ***USE VTITULURI IN 0***
O atentie deosebita merita modul de propagare a modificărilor operate în tabela derivată catre tabela principala din serverul de date Oracle –EDITURI. In mod general, o tabela derivata la distanta prezinta un mod de lucru de tip *otimistic Row Buffering,* aceasta reprezintă că modificarea unei linii se comite în tabela de baza , atunci căND pointerul VFP al tabelei derivate se pozţionează pe o alta inregistrare sau la executia functiei TABLEUPDATE(), pe de alta parte actualizarea tabelei derivate în BD Oracle se realizeaza prin functia REQUERY()
În dezvoltarea aplicatiilor este necesara preluarea şi tratarea erorilor provocate de catre tabela derivata bazei de date Oracle
Valoarea care se repeta pentru cheia primara ……
Problema :Cum preluam intr-o aplicatie VFP(Client), acest mesaj şi cum rezolvam problema fara combinatia :CTRL+ALT+DEL ?
Solutia vine de la funcţia AERROR().Aceasta plaseaza intr-un vector cu numele indicat.
Pentru erorile ODBC fiecare component al masivului contine :
Descrierea informatiilor obtinute prin functia AERROR()

Component	Tip si continut
1	Numeric :contine valoarea 1526 (connectivity error……..)indifernt de cauza eroriiODBC
2	Character:Textul mesajului de eroare
3	Character:Textul de eroare ODBC
4	Character:Starea curenta SQL ODBC
5	Numeric :Numarul erorii preluat de la sursa ODBC
6	Numeric:Numarul conexiunii ODBC
7	Valoarea NULL

De exemplu, daca la editarea unui căMp din tabela derivate vtitluiri , numele campului titlul incepe cu o litera mica, in momentul comiterii modificarii se genereaza o eroare care, preluată prin functia =AERROR(verr), iniţializeaza masivul verr, dupa cum urmeaza :
D*escrierea continutului masivului verr*

Componenta vErr	Continut
Verr(1)	1526
Verr(2)	Connectivity error:[Oracle][ODBC][Ora]ORA-02290:check constraint(BIBLIO.SYS_C004219)violated
Verr(3)	[Oracle][ODBC][Ora]ORA-02290:check

	constraint(BIBLIO.SYS_C004219)violated
Verr(4)	23000
Verr(5)	2290
Verr(6)	1
Verr(7)	NULL

In BD Oracle a fost definita, pentru atributul editura din tabela EDITURI, o restrictie prin clauza **check(editura=ltrim(initcap(editura))**, care trebuie să fie şiruri de caractere care incep cu majuscule si fara spatii.
In continuare vom prezenta cele mai importante proprietati, metode si evenimente de lucru cu baza de date Edituri , aflata pe serverul bazei de date Oracle

9.4.2 Proprietaţi,metode si evenimente ale formularului edituri

Definitie:Proprietatile sunt variabile locale ale fiecarui obiect, numite şi atribute, ce pot lua diverse valori, în conformitate cu funcţionalitatea acestora
Metodele sunt proceduri specifice asociate fiecarui obiect în parte, ce se executa în momentul invocarii lor explicite printr-o linie de cod dintr-o alta procedura
Prin intermediul evenimentelor si al metodelor (proceduri-eveniment) se implementeaza comportamentul obiectului.Proprietatile, metodele si evenimentele unui obiect sunt accesbile codului-sursa implementat la nivel superior
Sintaxa generala pentru a accesa o proprietate sau o metoda a unui obiect este urmatoarea :

Numecontainr.numeobiect.numeproprietate(=valoare)

Sau

Variavila=numecontainer.numeobiect.numeproprietate

sau numecontainer.numeobiect.numemetoda

In cazul nostru de faţă contianerul de nivel cel mai înalt este formularul, care in cod se specifica prin expresia THISFORM, respctiv in codul asociat unui eveniment, numele controlului se poate scrie cu THIS

Proprietati

Vom prezenta in continuare cele mai importante proprietati comune mai multor obiecte

1)value-stocheaza valoarea curenta a obiectului

Control	Posibile tipuri de date pentru proprietatea value
Checkbox	Integer,logical,numeric
Combo-box	Character,integer,numeric
Editbox	Characeter,memo
textbox	Orice tip

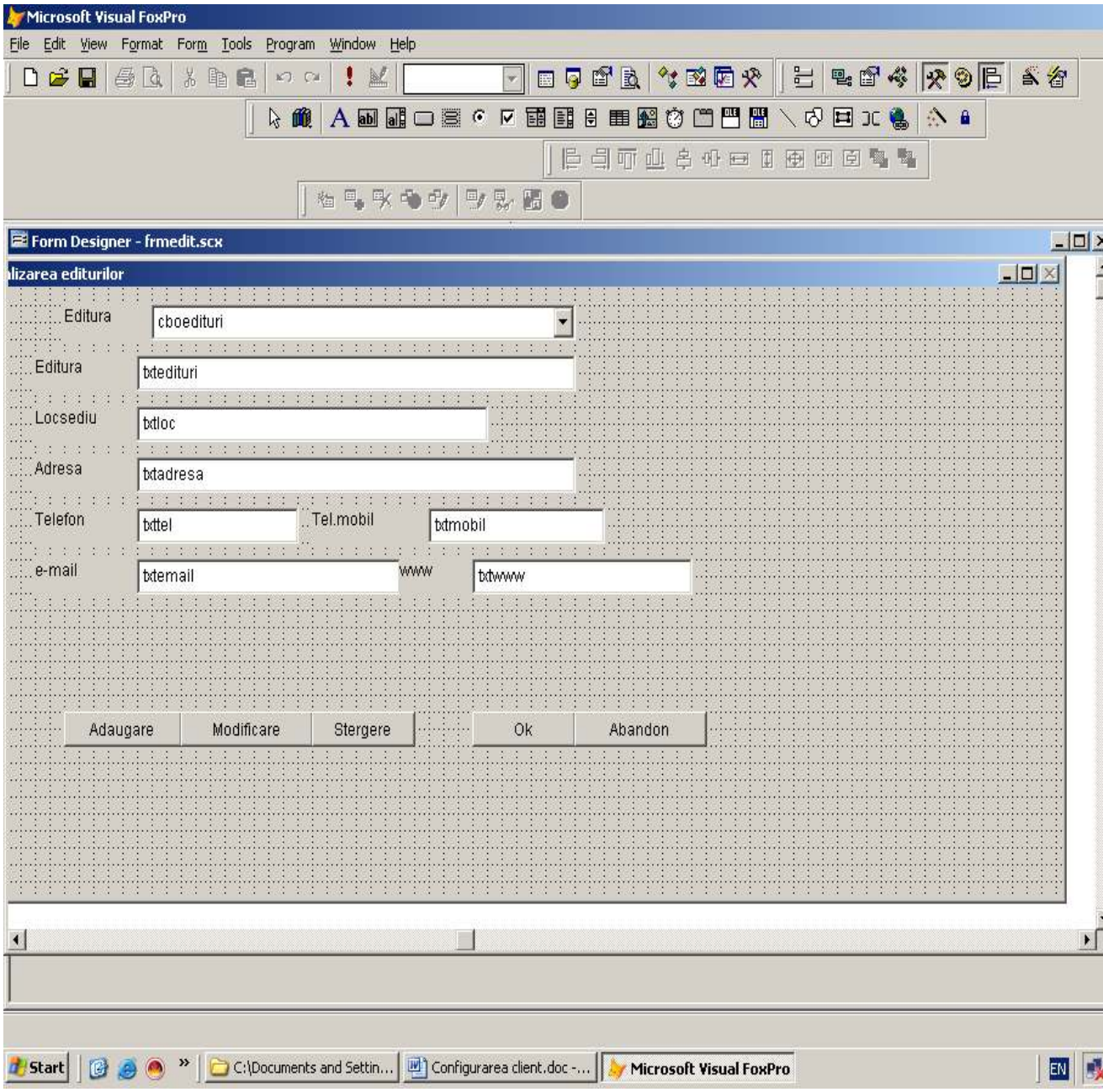
2)controlsource-specifica sursa de date a unui obiect legat
3)rowsource-(combo-box,listbox)-specifica sursa pentru elementele ce vor popula o lista
4)rowsourcetype-dedicata setarii tipului sursei elementelor unui obiect de tip lista

1-value	Pentru valori introduse direct in rowsource
2-alias	Specificarea campurilor unei tabele
3-SQL	Specificarea unei fraze SELECT-SQL
4-query	Specificarea unei proceduri de interogare ce creaza un cursor
5-array	Sursa de populare va fi formata din elementele unui tablou
6-fields	Lista este populata numai cu campurile specificate
7-files	Lista de fisiere din directorul curent
8-structure	Numele campurilor tabelei specificate in rowsource

9-popup	
10-	

5)columncount-specifica numarul de coloane ce va fi afisat intr-un obiect de tip lista
6)columnwidts-stabileste dimensiunea pentru coloanele unui obiect
7)boundcolumn-specifica a cata coloana este cea legata de sursa de date
8)recordsource-specifica sursa de date
9)recordsourcetype-tipul sursei de date pentru un control Grid
Evenimente si metode
Evenimentele si metodele folosite in proiectarea formularelor ar fi urmatoarele :

- Init-eveniment- se declanseaza in momentul in care se creaza un obiect, la lansarea in executie a uni formular
-
- Destroy se excuta in momentul in care un obiect este distrus
-
- Load –se declanseaza inainte de crearea oricarui obiect
-
- Release-metoda –specifica doar formularelor,se declanseaza la invocare efectiva (thisform.release)-inchiderea formularului
-
- Activate-eveniment-specific formularelor si controalelor de tip pageframe(cadru de pagina)
-
- Gotfocus-se declanseaza in momentul cand un obiect primeste controlul (focus-ul)-prin click
-
- Valid-se declanseaza inainte ca un obiect sa piarda controlul
-
- Setfocus,refresh-metoda-controlul va fi predat obiectului specificat/la invocare explicita printr-o linie de cod
-
- Requery-metoda-reevalueaza expresia din proprietatea rowsource si actualizeaza lista de elemente



Formular pentru actualizarea tabelei vedituri in faza de proiectare
Principalele obiecte ale formularului

Tip obiect	Nume obiect	Zona	Functionalitate
Textbox	Txtloc Txtadresa Txttelefon,txtmobil,txtemail,txtwww	2	Afisare/editare camp vedituri.editura Locsediu,telefon,adresa,mobil www
Combo-box	Cobedituri	1	Cuprinde lista editurilor din baza de date in ordine alfabetica.La selectia unei edituri se realizeaza pozitionarea cursorului pe inregistrarea corespunzatoare in tabela sursa şi nu este legat la nici

			o sursa de date
Command button	Cmdadg,cmdmod,cmdstg,cmdok,cmdabandon	3.	Butoane pentru actualizare,ok si abandon

Prefixe folosite pe parcursul aplicatiei

Tip control	Prefix	Tip control	Prefix
Check box	Chk	Option group	Opg
Command button	Cmd	Text box	Txt
Command group	Cgr	line	Lin
Container	Ctn	Grid	Grd
Edit box	Edt	Image	Img
Form	Frm	shape	Shp
Label	Lbl	Spinner	Spn
Option group	Opt	List box	Lst
Page frame	pfr	Timer	Tmr
Combo-box	Cbo,cmb		

Proprietati,metode si evenimenteale formularului edituri
Precizari privind controalele formularului frmedituri

Proprietate	Valoare	Explicatii
Control:txteditura		
Controlsource	Vedituri.editura	Controlul este legat la atributul editura din tabela derivate vedituri
Enabled	.F.-false	La lansarea formularului ,controlul este dezactivat
Selectonentry	.T.-true	La preluarea controlului in momentul executiei,continutul obiectului va fi implicit selectat,setare foarte utilă fiindcă utilizatorul trebuie să steargă conţinutul vechi şi apoi să introducă noua valoare
Tableindex	2	Indică numarul de ordine al obiectului,ce va determina la al catelea TAB ,enter va primi controlul
Control :txtloc		
Controlsource	Vedituri.locsediu	
enabled	.f.	
selectonentry	.t.	
tableindex	3	
Control:txtadresa		
Controlsource	Vedituri.adresa	
Enabled	.f.	
selectonentry	.t.	
Control:txttelefon	Este legat la sursa de date vedituri.telefon	

Control:txtmobil	Vedituri.mobil	
Control:txtemail	Vedituri.email	
Control:txtwww	Vedituri.www	
Control:cboedituri		
controlsource	(none)	Are rolul de pozitionare pe o editura si de improspatare a controalelor ce afiseaza continutul inregistrarii curente,daca nu este legat la sursa de date ,proprietatea value va stoca valoarea curenta de pe coloana specificata in proprietatea boundcolumn
enabled	.t.-true	
tableindex	1	
rowsourcetype	6-Fields	Lista este populate numai cu valorile campurilor specificate,utilizand formatul:NUME_TABELA.CAMP2,CAMP1…..
Rowsource	Vedituri.editura	
boundclomun	2	A câta coloana din cele furnizate de controlul rowsource este cea legata la controlsource
columncount	2	Numarul de coloane afisate la deschiderea listei(click,sau space sau alt+sageti)
columnwidts	200,150	Latimea fixa a coloanelor ce vor fi afisate pentru aspectul estetic al listei
columnlines	.t.	Linii care separa coloanele in partea desfasurata a listei
style	2-dropdownlist	Proprietatea poate avea: 0-dropdwncombo-utilizatorul va avea posibilitatea fie să aleagă un element din lista,fie să-l introduca 2-dropdwnlist-nu va exista decat posibilitatea alegerii unui element din lista,in care se tasteaza primele 2-3 caractere,iar proprietatea incrementserarch=.t.
Control:cmdadg,cmdmod……		Pentru actualizarea bazei de date
Obiect:frmedit(formularului principal)		
caption	Actualizarea editurilor	Text ce apare ca titlu al ferestrei formularului
windosstate	1-modal	Utilizatorul nu va avea acces la meniul principal al aplicatiei,sau in alt formular decat dupa iseirea din acest

		formular

Surse de date ale controalelor
Avem doua posibilitati de rezolvare a problemei:
a)fie utilizam sursa de date a formularului –data environment-ce se intializaeaza inaintea obiectelor;
b)fie introducem instructiuni de deschidere a tabelelor in evenimentul Load al formularului

Metode ale formularului frmedit
Obiect frmedit
Listing 1 Metoda init a formularului frmedit
****
```
Public verr(7),a
If !used('vedituri')
Use vedituri in 0 exclusive
Endif
Select vedituri
Requery()
If eof() or bof()
    Go top
Endif
Thisform.boeditura_refresh
Thisform.dezactivare
```

Metoda cboedituri_refresh invocate este cea care afiseaza in combo-box cboedituri valoarea curenta din veditura ale campului editura
Listing 2Metoda cboedituri_refresh
****
```
For i=1 to thisform.cboeditura.listcount
    If alltrim(thisform.cboeditura.listitem(i))==
Alltrim(veditura.editura)
Thisform.cboeditura.listindex=i
Exit
Endif
Endfor
```

Metoda asociata evenimentului click a butonului OK(cmdok) este cea care declanseaza transmiterea modificarilor din tabela derivate vedituri in tabela de baza din Oracle Edituri
Listing 3 Metoda click a butonului cmdok
****
```
Thisform.scrie_buffer
If !a
Thisform.txteditura.setfocus
Else
```

```
pozitie_=recno()
sele vedituri
requery()
thisform.cboeditura.requery()
go pozitie_
thisform.cboeditura_refresh
thisform.refresh
thisform.dezactivare
endif
```

listing 4 Metoda dezactivare – care ce creaza cu ajutorului meniului Form-New Method

```
**********************
thisform.cboeditura.enabled=.t.
thisform.txteditura.enabled=.f.
thisform.txtloc.enabled=.f.
thisform.txtadresa.enabled=.f.
thisform.txttelefon.enabled=.f.
thisform.txtmobil.enabled=.f.
thisform.txtemail.enabled=.f.
thisform.txtwww.enabled=.f.
thisform.cmdadg.enabled=.t.
thisform.cmdmod.enabled=.t.
thisform.cmdstg.enabled=.t.
thisform.cmdok.enabled=.t.
thisform.cmdabandon.enabled=.t.
```

listing 5 Metoda activare –care se creaza la fel Form-New Method

```
**********************
thisform.cboeditura.enabled=.f.
thisform.txteditura.enabled=.t.
thisform.txtloc.enabled=.t.
thisform.txtadresa.enabled=.t.
thisform.txttelefon.enabled=.t.
thisform.txtmobil.enabled=.t.
thisform.txtemail.enabled=.t.
thisform.txtwww.enabled=.t.
thisform.cmdadg.enabled=.f.
thisform.cmdmod.enabled=.f.
thisform.cmdstg.enabled=.f.
thisform.cmdok.enabled=.f.
thisform.cmdabandon.enabled=.f.
```

In continuare voi prezenta cea mai importanta metoda, anume metoda **scrie_buffer** care incearca sa scrie continutul bufferului VFP, inregistrarea modificata a tabelei derivate vedituri in masivul verr cu informatii despre eroare.Creem o tabela temporara pentru jurnalizarea erorilor

CREATE TEMP(NR INTEGER,TEXT CHAR(250))

Listing 6 Metoda scrie_buffer –la fel se creaza din meniul Form –New Method

```
****************************
Select vedituri
Release all like sir*
a=tableupdate()
If !a
```

```
=aerror(verr)
For i=3 to 7  && conversia in sir de caractwere a utimelor cinci componente ale vectorului
I1=str(I,1)
Sir&i1=NVL(VERR(I),”NULL”)
Sir_=sir&i1
If type(“sir_”)=”N”
   Sir&i1=str(sir&i1,10)
Endif
Endfor0
Insert into temp values(1,str(verr(1),7))
Insert into temp values(2,verr(2))
Insert into temp values(3,sir3)
Insert into temp values(4,sir4)
Insert into temp values(5,sir5)
Insert into temp values(6,sir6)
Insert into temp values(7,sir7)
Do case
     Case verr(1)=1585 && update conflict
    Messagebox(‘inregistrarea actualizate a fost modificata intre timp de altcineva!’)
     Case verr(1)=1526 and verr(5)=1 and “pk_editura”$verr(2)
 Messagebox(‘se repeat valoarea atributului editura !’)
      Case verr(1)=1526 and verr(5)=1 and nn_editura” $verr(2)
Messagebox(“se repeat valoarea atributului editura’)
       Case verr(1)=1526 and verr(5)=2290 and  ck?editura $ verr(2)
Messagebox (“valoarea aributului editura depaseste limita stabilita “)
Case verr(1) =1526 and verr(5)=20015
Messagebox(verr(3),31,65))
Otherwise
Messagebox(“eroaree netrata !’)
Endcase
=tablerevert()
Endif
```

Alte metode ale formularului frmedit

Listing 7 Metoda cmdabandon

```
**************************
Tablerevert(.t)
Thisform.release
```

Listing 8 Metoda asociata evenimentului cmdadg && adaugare de articole

```
*****************************************
Sele vedituri
Append blank
Thisform.scrie_buffer
Requery()
Go bottom
Thisform.cboeditura_refresh
Thisform.refresh
Thisform.activare
```

Listing 9 Metoda asociata evenimentului cmdmod –modificarea articolelor

```
**************************************
Thisform.activare
```

Listing 10 Metoda asociata evenimentului cmdstg-stergere de articole

```
**************************************
Select vedituri
Delete
Thisform.scrie_buffer
If a
   If eof()
      Go bottom
  Else
       If bof()
            Go top
       Endif
Endif
pozitie_=recno()
Requery()
Thisform.cboeditura.requery()
Go min(pozitie_,recount())
Thisform.cboeditura_refresh
Thisform.refresh
Endif
**********************************************
```

9.4.3Tehnologia SPT(SQL Pass-trough)[7]

Cea de-a doua modalitate de a lega aplicatii Visual Foxpro la servere de baze de date Oracle o constituie tehnologia ***SQL Pass-Trough(SPT).***
Dezavantajul, prin comporaţie cu tabele derivate la distanta, ţine de faptul ca efortul este foarte mare de lucru, iar avantajul tine de flixibilitate.
Interogarile SPT creeaza implicit un cursor anume o copie neactualizabila a dateleor extrase din BD Oracle, cursorul poate fi actualizabil prin setarea parametrilor cu ajutorul funcţiei ***CURSORSETPROP().***

1)Stabilirea conexiunii

In prima etapa trebuie rezlizată stabilirea unei conexiuni ODBC.Legarea la BD se realizeaza prin functia SQLCONNECT() sau SQLSTRINGCONNECT().Prin SQLCONNECT() , fara argumente se afiseaza o fereastra de dialog prin care utilizatorul îşi poate alege una dintre sursele de date, iar SQLDISCONNCET()-dezactiveaza conexiunea cu BD aflata pe server
Variabila nrconexiune , va conţine numarul de conexiunii stabilite

Exemplu 1 test_spt_1.prg

```
**********************
Nrconxiune=SQLCONNECT('Biblio')
IF nrconexiune<1
=messagebox(« Conexiunea esuata »,0, »Reaultat tentativa »)
Return
Endif
```

2)Interogarea BD si apelul procedurilor stocate Oracle

Pentru a accesa date aflate pe serverul BD Oracle, este necesara declararea unui canal/magistrala pentru conexiunea active.Legatura se poate face din VFP cu ajutorul functiei SQLEXEC() ,care pot fi lansate comenzi Oracle DML si DDL si pot fi apelate proceduri stocate

Exemplu 2 test_spt_2.prg din VFP –prin fraza SELECT se interogheaza tabela Edituri din Oracle

```
Vsucces=SQLEXEC(nrconexiune,'SELECT * FROM EDITURI','CEDITURI')
IF VSUUCES>0
SELECT cedituri
Browse
Else
=mesasagebox("interogarea fara rezultat",0,"rezultatul interogarii")
Endif
```

Variabila *vsucces* va contine numarul de seturi de rezultate furnizat de serverul BD Oracle (in mod general avem 1 sau 0 daca functia este in curs de executie), iar setul de rezultate furnizat de Oracle va deveni in VFPun cursor denumit **Cedituri**
In dezvoltarea aplicatiilor,pentru optimizarea traficului pe reţea , este forate important ca interogarile sa fie parametrizate

Exemplu test_spt_3.prg

****

```
Editura_= « All »
Vsucces=SQLEXEC(nrconexiune,'SELECT * FROM Edituri where
editura= ?editura_','cedituri')
Select cedituri
Browse
```

Variabila –parametru *editura*_este folosita pentru a extrage din tabela Oracle EDITURI numai inregistrarile cu editurile din tabela, evident poate fi folosita in formulare
Invocarea unei proceduri stocate de pe serverul BD este una dintre cele mai importante facilitate ale tehnologiei SPTprin compratie cu tabele derivate la distanta

.

Exemplu Stegerea unor inregistrari din tabela EDITURI aflata in BD Oracle, nu prin tabele derivate, nici prin intermediul comenzii DELETE –SQL,, ci invocand o procedura stocata care va fi executata pe serverul Oracle, crearea se face folosind utilitarul SQL*PLUS, respectiv conectat la schema **biblio**

Exemplu 4 test_spt_4.sql

****

```
CREATE OR RREPLACE
Procedure STERG_EDITURA
(PEDITURA IN CHAR,prezultat OUT varchar2)
Is
Violare_fk exeception;
V_locsediu number;
Pragma exception_init(violare_fk,-02292);
BEGIN
    DELETE FROM EDITURI WHERE editura=peditura;
```

```
Prezultat:=’succes’;
EXCEPTION
WHEN violare_fk THEN
SELECT EDITURA INTO V_locsediu FROM EDITURI WHERE
EDITURA=PEDITURA;
DELETE FROM EDITURI WHERE EDITURA=PEDITURA;
Prezultat:=’a fost starsa editura cu locsediueditura ‘||v_locsediu;
When others then
Prezultat:=’insucces’;
END;--Procedure sterg_edituri
```

Invocarea unei procedurii stocate se face in mod obisnuit in felul urmator:
=SQLEXEC(nrconexiune,”BEGIN nume_procedura_stocata;END;”)

Oproblema ridicata de apelul procedurilor si functiilor stocate Oracle o constituie transmiterea paramterilor si preluarea rezultatelor returnate de functii

Exemplu : test_spt_biblio.prg

```
******************************
*****de invocarea unei proceduri stocate Oracle cu parametric de intrare
**************************************************
Nrconexiune=SQLCONNECT(‘biblio’)
If nrconexiune<1
=messagebox(“conexiune esuata”,0,”rezultat tentative”)
Return
Endif

****preluarea valorilor variabilelor-parametri ai procedurii ORACLE
Veditura=’editura’
Vrezultat=’incert’
Nrezultat=SQLEXEC(nrconexiune,”BEGIN
sterg_editura(?veditura,?@vrezultat) ;END ; « )
If nrezultat<0
Messagebox(« problema »)
DIME veroare(1,1)
=aerror(veroare)
Else
Messagebox(vrezultat)
Endif


=SQLDISCONNECT(nrconexiune)
Nrconexiune=SQLCONNECT(‘biblio’)
If nrconexiune<1
=messagebox(« conexiunea n-a mai putup fi restabilita ! »,0, »rezultat tentativa »)
Return
=SQLEXEC(nrconexiune, »SELECT * FROM EDITURI’,’CEDITURI’)
SELECT CEDITURI
BROWSE
```

Exemplu de preluare in variabile VFP a rezultatului unei functii PL/SQL

Variabila veditura a fost intializata in proframul VFP avand ca scop transmiterea valorii editura vizat procedurii stocate de pe serverul Oracle prin intermediul apwelului ***SQLEXEC()***, respectiv rolul variabilelei vrezultat a fost de a prelua valoarea parametrului de iesire al procedurii sterg_editura.
In mod natural aceste variabile sunt legate la controalele formularului frmspt_editura.scx, iar în urma executiei este posibil ca executia să se soldeze cu erori din partea BD oracle, pentru aceasta exista o procedura pentru rezolvarea sa prin functia ***AERROR()***
SQLDISCONNECT() si secventa care urmeaza are drept scop reimprospatarea cursorului
Sintaxa folosita pentru transmiterea valorii unei variabile locale in apelul unei proceduri stocate *(?nume_variabila) si pentru preluarea valorii returnate de o procedura stocata printr-un parametru de iesire(?@nume_variabila*
Exemplu pentru crearea functiei tel_edituri
Listing test_spt_functie.sql

```
**********
Create or replace  function tel_edituri (peditura  in varchar2, padresa in
varchar2,ptel out varchar2)
Return varchar2
Is vtelefon varchar2(100);
Begin
Select count(*) into vtelefon ,peditura from edituri where  editura=peditura and
adresa=padrea;
Return vetelefon;
End;
/
```

Programul VFP ce apeleaza si preia rezultatul acestei functii PL/SQL este :
Listing test_spt_functie.prg

```
**********
Nrconexiune=SQLCONNECT('BIBLIO')
IF NRCONEXIUNE<1
=MESSAGEBOX(« CONEXIUNE ESUATA »,0, « ABANDON »)
RETURN
ENDIF

PEDITURA='ALL'
PADRESA='STRADA….'
VTELEFON=''
NREZULTAT=SQLEXEC(NRCONEXIUNE,'DECLARE R INTEGER ;BEGIN
R :=TEL_EDITURI( ?PEDITURA, ?PADRESA,?@PTELE);END;')
IF NRREZULTAT<0
****OPERATIUNE ESUATA
MESSAGEBOX('EROARE')
DIME VEROARE(1,1)
=AERROR(VEROARE)
*SECVENTA DE ERORI

ELSE
MESSAGEBOX('EDITURA'+PEDITURA+CHR(13)+PADRESA)
ENDIF
ENDIF
```

Funtia ***VFP SQLEXEC() c***e apeleaza functia Oracle contine un mininloc PL/SQL, in esenta preluarea rezultatului din Oracle este variabila vtelefon ce trebuie marcata in *?@nume_variabila*

9.4.4.Cursoare Actualizabile

In dezvoltarea aplicatiilor VFP-Oracle mai este important si modul de propagare a modificarilor operate in tabele/cursoarelor VFP in baza de date aflata pe server, care se face cu ajutorul funcţiei **CURSORSETPROP().**

Exemplu de creare a unui cursor VFP actualizabil

Listing 9.4.4_cursor_actualizabil_edituri.prg

****

```
#include foxpro.h
Nrconexiune=SQLCONNECT('biblio')
If nrconexiune<1
=messagebox("conexiune esuata",0,"abandon")
Return
Endif
If uesd('cedituri')
Sele cedituri
Use
Endif

Vsucces=SQLEXEC(nrconexiune,'SELECT * FROM edituri','cedituri')
If vsucces<1
=messagebox("interogare fara rezultat",0,"rezultatul interogarii")
Return
Endif

Set multilocks on
*se declara bufferingul de tip optimitisc
=CURSORSETPROP("Buffering",3,"cedituri")
=CURSORSETPROP('Tables','edituri','cedituri')
=CURSORSETPROP('Keyfieldlist','editura','cedituri')
=CURSORSETPROP('updatablefieldlist','editura,locsediu,adresa,telefon,mobil,e-
mail,www','cedituri')
*maparea atributelor cursorului la atributele tabele de baza
=CURSORSETPROP('updatenamelist','editura edituri.editura,locsediu
edituri.locsediu,adresa,; edituri.adresa,telefon edituri.telefon,mobil
edituri.mobil,email edituri.email,;
 edituri.www','cedituri')
=CURSORSETPROP('WhereType',DB_KEY,'Cedituri')
=CURSORSETPROP('Sendupdates',.T.,'cedituri')
=CURSORSETPROP('UpdateType',DB_UPDATE','cedituri')
```

Din momentul declararii si editarii cursorului, modificarile operate vor fi propagate automat in baza de date **Biblio** de pe serverul Oracle.

9.4.5 Formulare VFP-accesul si modificarea datelor aflate pe serverul Oracle

Pentru ca un formular VFP sa lucreze cu date stocate in serverul Oracle,trebuie supus unui process de transformare ,anume:

1)definirea si redefinirea mecanismelor de acces la inregistrarile din tabela(tabelele) de baza stocate pe serverul Oracle
2)redefinirea sursei de datelor pentru obiectele de acces la date ale formularului
3)redefinirea mecansmelor de navigare pe inregistrarile accesibile prin formular
4)redefinirea mecanismelor de control al operatiilor de actualizare(adaugare,modificare,stergere)
5)redefinirea unui mecanism de tratare a erorilor pentru operatiile neincheiate cu success

Exemplu fie formularul proiectat anterior frmedit:**Actualizarea editurilor**
Transformarea formularului se face astfel:
a)Sursa de date a formularului :

- Daca se lucreaza cu dataenvironment, atunci tabela **EDITURI** din baza de date locala VFP va fi scoasă şi înlocuita cu tabela derivata la distanta **vedituri**
- Daca tabelele de baza sunt gestionate in metoda Load a formularului , atunci in codul-sursa al acesteia va fi specificata instructiunea **USE VEDITURI IN 0**

Listing 1 METODA Load a formularului frmedit in varianta SPT

```
**********************************************************
Local nrconexiune,exista_tag
If not dbused('biblio')
Open database '\biblio\database\biblio" shared
Endif

**se deschid tabela derivate vedituri
If !used('vedituri')
Use vedituri in 0
Endif

**se creeaza cheia de navigare necesara pentru indexare
Select vedituri
For i=1 to tagcount()
If alltrim(upper(tag(i)))='editura'
Exista_tag=.t.
Endif
Endfor
If !exista_tag
Index on editura tag editura
Endif
***se stabilesc sursa de date pentru cboeditura
Select vedituri
Nrconexiune=cursorgetprop('connecthandle')
vsucces=SQLEXEC(nrconexiune, »select
rtrim(ltrim(editura))||''||rtrim(ltrim(adresa)) ,;
editura from edituri  order by editura,adresa","crsedituri")

if vsuuces<0
messagebox('eroare !nu s-a reusit popularea editurii')
return .f.
endif
```

b)sursele de date pentru obiectele de acces:

- Pentru căsuţele txteditura, txtadresa, txttelefon, txtmobil, txtemail, txtwwww proprietatile valorilor vor fi **vedituri.edituri** etc…….
- Pentru lista combinata cboeditura, proprietatea rowsourcetype=alias, rowsource=crsedituri, care secreaza in fraza SQLEXEC
- Navigarea in formularul frmedit se realizeaza prin intermediul listei cboeditura

c)mecanismul transactional (BEGIN TRANSACTION-….END TRANSACTION) şi operatiile de inserare/modificare/stergere raman la fel ca in cazul local

LISTING 2 Metoda Clik a butonului cmdok

****

```
END TRANSACTION
Select vedituri
Thisform.scrie_buffer                && metoda prezentata anterior
Thisform.refresh
Thisform.dezactivare                 && Metoda prezentata anterior
Thisform.cboeditura.requery
For i=1 to thisform.cboeditura.listcount
  If thisform.cboeditura.list(I,2)==veditura.editura
     Thisform.cboeditura.listindex=i
Endif
Endfor
```

Listing 3 Metoda destroy a formularului frmedit

****

```
If txnlevel()>0
If messagebox('salvati utimele modificari ?',32+4,'utimele nu au fost salvate')=6
End transaction
Else
Rollback
Endif
Endif
If thisform.asters.t.
Select edituri
Thisform.scrie_buffer
Endif
```

Listing 4 Metoda clik de adaugare a butonului cmdadd

****

```
Begin transaction
Insert into vedituri values(editura_,locsediu_,adresa_,telefon_,'','','')
thisform.scrie_buffer
thisform.activare
thisform.txteditura.SetFocus
```

listing 5 metoda clik de modificare de articole cmdmod

****

```
BEGIN TRANSACTION
thisform.activare
thisform.txteditura.SetFocus
```

listing 6 metoda clik de stergere de articole cmdstg
****

```
SELECT vedituri
IF MESSAGEBOX('sunteti sigur ?',32+4,'clasa va fi
stearsa !!!')=6
    DELETE
    thisform.asters=.t.
thisform.scrie_buffer
ENDIF
```

Proprietatea *asters* se creaza folosind meniul Form –New Proppriety
Listing 7 Metoda clik a cmdabandon-abandon
****

```
=tablerevert(.t.)
Thisform.release
```

d)conversia rapoartelor astfel incat sa se foloseasca sursele de date stocate pe serverul Oracle
e)traducerea frazelor SELECT-SQL specifice VFP in sintaxa Oracle si executarea acestora folosind instructiunea SQLEXEC

9.4.6 Solutii sql si Oracle *Reguli,incluziuni si cheie surogat.aplicatii practice*

1.chei surogat

Definiţie
O cheie surogat este o cheie artificială sau sintetica utilizată ca substitut al unei chei naturale
Exemple de chei surogat :codcl(cod clientului), codpr(cod produs), idprof(codul profesorului)etc…
2.probleme privind cheiele surogat
Pentru Codd, cheiele primare definite şi controlate de utilizator, folosite ca surogate permanente pentru entităţi,ar prezenta trei difucltăţi la intrebuinţare :

- Valorile cheilor surogat trebuie uneori schimbate.de exemplu, atunci doua companii fuzioneaza, angajati trebuie reuniti intr-o singură tabela şi nu este exclus ca, din pricina suprapunerilor, unele mărci să fie modificate.
- Doua relatii ar putea avea chei surogat definite de utilizator pe domenii diferite chiar daca entitatile pe care le identifica sunt aceleasi ;
- De exemplu o firmă poate fi client şi furnizor al companiei noastre.
- Uneori este necesare preluarea informatiei despre o entitate inantea atribuirii de catre utilizator a unei chei surogat pentru aceasta sau, pe de alta parte pastrarea valorii cheii surogat chiar si dupa ce entitatea nu mai constituie subiect al aplicatiei.

Exemplu în Visualfoxpro 9.0
Un atribut poate fi declarat cheie surogat folosind clauza AUTOINC

CREATE TABLE CARTI(;
IDNRCT INTEGER AUTOINC NEXVALUE 10 STEP 1 PRIMARY KEY ;
)
Clauza aditională *nextvalue* este utila atunci cand se doreste ca valoarea initiala să nu fie 1,STEP stabileste marimea incrementată
În Oracle, presupune folosirea secventelor, care sunt obiecte ale bazei de date ce furnizeaza la fiecare invocare a clauzei **nextval** o valoare unica
Listing 9.4.6 secventa_1.sql
****
Sql>CREATE SEQUENCE seq_idnrct start with 10 minvalue 10 maxvalue 999999999 nocache nocycle order
Sevcenta seq_idnrct create va furniza valori ordonate strict după momentul apelului, cuprinse intre[10,999999999], iar după atingerea limitei superioare secvenţa se blocheaza(clauza nocycle), iar in lipsa clauzei , dupa atingerea valorii maxime , urmatoarea valoare furnizeazata este cea minima.
Declansatorul Oracle pentru valoarea implicita a cheii surogat

Listing trg_exemplare_ins.sql
****
Sql>CREATE OR REPLACE TRIGGER trg_exemplare_ins
BEFORE INSERT ON exemplare FOR EACH ROW
BEGIN
SELECT seq_idnrct.nextval INTO :new.IDNRCRT FROM dual;
END;

Exemple de chei surogat pentru tabela inventar de carti
Listing Creare_seq_inventar.sql
****
```
drop sequence seq_nrinv;
create sequence seq_nrinv INCREMENT BY  1
minvalue 1 maxvalue 999999
NOCYCLE NOCACHE ORDER ;

create or replace trigger trg_inventar_ins
before insert on inventar for each row
begin
select seq_nrinv.nextval into :new.nrinv from dual;
end;
/
```

Listing Creare_seq_exemplare.sql
****
```
drop sequence seq_idcota;
create sequence seq_idcota start with 1
minvalue 1 maxvalue 99999999999999
nocycle nocache order;

create or replace trigger trg_exemplare_ins
before insert on exemplare for each row
begin
select seq_idcota.nextval into:new.idcota from dual;
end;
```

/

Problema :*Grupul Scolar stabileşe urmatoarea restricţie : un manual nu poate fi achiziţionat în mai mult de 100 de exemplare !.Cum se poate implementa aceasta restrictie ?*
In baza de date *exemplare*(*cota,isbn)*-aceasta restrictie presupune ca la actualizarea bazei de date (inserare,modificare),,sa se verifice daca numarul cotelor pentrul noul ISBN (valoarea ISBN de linia inserata/modificata să nu depaseasca 100, situatie in care inserarea/modificarea sa fie blocata

Pentru aceasta se foloseste declansatoarele de inerare, modificare ale tabelei **Exemplare**
Declansator Oracle de inserare pentru implementarea restrictiei legate de numarul de exemplare dintr-o carte

Listing 4b.sql
****

```
SQL>CREATE OR REPLACE TRIGGER trg_exemplare_ins
        BEFORE INSERT ON exemplare
         FOR EACH ROW
DECLARE
          V_nrexemplare NUMBAER(5):=0;
BEGIN
       SELECT COUNT(*) INTO v_nrexemplare
        FROM exemplare WHERE ISBN=:NEW.isbn;
        IF v_nrexemplare>99 THEN
           RAISE_APPLICATION_ERROR(-20801,'Nr.exemplarelor
acestei carti creste peste 100!);
        END IF;
END;
```

In continuare cream o functie f_nrexemplare,careia I se plaseaza un ISBN si care returneaza numarul de exemplare al acelui ISBN,in tabela TITLURI,in care trebuie sa introducem atribul nrexemplare astfel:
ALTER TABLE TITLURI add NREXEMPLARE number(4);

Functia Oracle ce returneaza nr.de exemplare ale unei carti
Listing functia_exemplare.sql
****

```
CREATE OR REPLACE FUNCTION
f_nrexemplare(isbn_titluri.isbn%TYPE)
        Return number
IS
  V_nr NUMBAER(4):=0;
BEGIN
       SELECT nrexemplare INTO v_nr FROM titluri WHERE isbn=isbn_;
```

```
        RETURN v_nr;
END;
```

Declansatoarele tabelei EXEMPLARE care folosesc functia,se prezinta astfel:

-Pentru inserare

```
CREATE OR REPLACE TRIGGER trg_exemplare_ins
BEFORE INSERT ON exemplare
FOR  EACH ROW
BEGIN
    IF f_nrexemplare(:NEW.ISBN)>3 then
        RAISE_APPLICATION_ERROR(-2801,”NR.EXEMPLARE creste
peste 100!’);
ELSE
 /*SE incrementeaza nr.exemplarelor pentru isbn din linia inserata */
UPDATE titluri SET NREXEMPLARE=NREXEMPLARE+1 WHERE
ISBN=:NEW.isbn;
END IF;
END;
```

-pentru modificare

```
CREATE OR REPLACE TRIGGER trg_exemplare_upd
BEFORE INSERT ON exemplare
FOR  EACH ROW
BEGIN
    IF f_nrexemplare(:NEW.ISBN)>3 then
        RAISE_APPLICATION_ERROR(-2801,”NR.EXEMPLARE creste
peste 100!’);
ELSE
 /*se scade cu 1 nr.exemplarelor pentru isbn din linia inserata */
UPDATE titluri SET NREXEMPLARE=NREXEMPLARE-1 WHERE
ISBN=:OLD.isbn;

/*se incrementeaza nr.exemplarelor pentru isbn din linia inserata */
UPDATE titluri SET NREXEMPLARE=NREXEMPLARE+1 WHERE
ISBN=:NEW.isbn;
END IF;
END;
```

-pentru stergere

Declansatorul pentru stergere a unei linii din tabela EXEMPLARE

```
CREATE OR REPLACE TRIGGER trg_exemplare_del
BEFORE INSERT ON exemplare
FOR  EACH ROW
BEGIN
   /*se scade cu 1 nr.exemplarelor pentru isbn din linia inserata */
```

UPDATE titluri SET NREXEMPLARE=NREXEMPLARE-1 WHERE ISBN=:OLD.isbn;
END;
Oalta solutie mai rapida ar fi introducerea o regula de validare la nivel de atribut ,dupa cum urmeaza:
Sql>alter table titluri add constraint ck_nrexemplare
Check(nrexemplare<=100);

Exemple de cereri

1.Se considera baza de date BIBLIOTECA ,cu urmatoarele relatii
TITLURI*(isbn,editura,*titlu,anaparitie)
EDITURI(*editura*,locsediu,adresa)
Titluri_autori(isbn,autor)
Clienti(nrpermis,nume,adresa,localitatea)
Fise(nrpermis,isbn,dataimp)
Imprumut(nrpermis,isbn,nrinv,dataimp,autor,titlu,cota,numepren,datarest)

Cerere: listarea titlurilor cartilor care au fost imprumutate inainte de 01 noiembrie 2006
Rezolvare :Se ralizeaza o vedere asupra relatiilor
TITLURI,TITLURI_AUTORI,CLIENTI,FISE si se obtine tabela IMPRUMUT
$\Pi_R(\sigma_F$(FISE*CLIENTI*TITLURI_AUTORI*TITLURI))
unde R este relatia ce contine atributele
isbn,editura,titlu,autor,nrpermis,numepren,locsediu,adresa,localitatea,anaparitie
F= este conditia *clienti.nrpermis=fise.nrpermis and titluri.isbn=fise.isbn*
In cazul de fata avem :$\Pi_{TITLU}(\sigma_{DATAIMP<01/11/2006}$(IMPRUMUT))

2.Să se listeze cartile imprumutate ,presupunand ca data restituirii este de 14 zile,in perioada 15-30 noiembrie 2006
S*olutia 2.1 i*n Visual Foxpro este urmatoarea :

```
SELECT DISTINCT nrpermis,numepre,titlu,autor,isbn,cota,dataimp,;
dataimp+14 as restituire    from imprumut;
where  dataimp+14 >={^2006/11/15} AND dataimp+14 <={^2006/11/30}
```

solutia 2.2 folosind operatorul BETWEEN

```
SELECT DISTINCT nrpermis,numepre,titlu,autor,isbn,cota,dataimp,;
dataimp+14 as restituire    from imprumut;
where  dataimp+14 BETWEEN{^2006/11/15} AND {^2006/11/30}
```

3.Restituirea fiecarei carti este de 14 zile.Daca insa data-limita cade intr-o sambata sau duminica atunci restituirea se muta in lunea urmatoare.Care sunt zile restituite in aceste conditii ?
Solutia 3.1 in VFP
VISUAL FOXPRO prezinta functiile CDOW(CHARACTER DAY OF THE WEEK) si DOW
Listing 3.1.restituirea_VFP.PRG

```
SELECT nrpermis AS permis,DATAIMP,;
DATAIMP+14 AS RESTITUIREA1,;
CDOW(DATAIMP+14) AS Numezile1,;
Iif(DOW(DATAIMP+14)=6,;
```

```
  DATAIMP+16,;
   IIF(DOW(DATAIMP+14)=1,;
      DATAIMP+15,;
       DATAIMP+14) AS RESTITUIREA,;
CDOW(IIF(DOW(DATAIMP+14)=6,;
DATAIMP+16,;
IIF(DOW(DATAIMP+14)=1,;
DATAIMP+15,;
DATAIMP+14)) AS ZI_RESTITUITA;
FROM IMPRUMUT
```

SOLUTIA 3.2 ORACLE ,se utilizeaza structura CASE si functia TO_CHAR

Listing solutia3.2_restituirea_oracle.sql

```
SELECT NRPERMIS,TO_CHAR(DATAIMP,'YYYY-MM-DD') AS DATAIMP,
TO_CHAR(DATAIMP+14,'YYYY-MM-DD') AS RESTITUIREA1,
TO_CHAR(DATAIMP+14,'DAY') AS NUMEZILE1,
CASE WHEN TO_CHAR(DATAIMP+14,'DAY')='DATURDAY'
     THEN TO_CHAR(DATAIMP+16,'YYYY-MM-DDD')
 ELSE
CASE WHEN TO_CHAR(DATAIMP+14,'DAY')='SUNDAY'
      THEN TO_CHAR(DATAIMP+15,'YYYY-MM-DD')
     ELSE  TO_CHAR(DATAIMP+14,'YYYY-MM-DD')
END
END AS RESTITUIREA,
TO_CHAR(CASE WHEN TO_CHAR(DATAIMP+14,'DAY')='SATURDAY'
  THEN DATAIMP+16
   ELSE
      CASE WHEN TO_CHAR(DATAIMP+14,'DAY')='SUNDAY'
          THEN DATAIMP+15
             ELSE    DATAIMP+14
       END
END,
'DAY') AS  zi_restituire
FROM  IMPRUMUT;
```

9.4.7 Instantanee[10]

Conceptual de instantaneu (**snapshot)** este strâns legat de conceptele de baza de date distribuita şi replicarea unei baze de date.

Practic pe fiecare calculator exista un server Oracle care gestioneaza baza de date locala dar coopereaza şi la menţinerea consistentei datelor din celelalte baze de date situate la distanta, asigurând astfel functionarea unei bazei de date distribuite (bdd)

Prin urmare o aplicaţie poate accesa şi modifica prin intermediul unei reţele date conţinute în mai multe baze de date, acestea fiind vazute ca o baza de date singulara ce are proprietatea de a avea datele distribuite pe mai multe calculatoare.

Replicarea unei baze de date reprezintă facilitatea de a putea copia datel dintr-o baza de date in locatii multiple si sisteme situate la distanta.
Replicarea unei baze de date poate fi facuta la nivelul intregii baze de date sau la nivelul unor anumite tabele,prezentate in capitolul precedent
Instantaneul este un tabel care contine rezultatele unei interogari asupra unuia sau mai multor tabele care in general se regasesc intr-o baza baza de date situata la distanta.
Tabele comune in interogare se mai numesc *tabele master* iar baza de date ce contine aceste tabele se numeste *baza de date master*.Interogarea pe care bazeaza instantaneul nu poate contine tabele sau vederi care au ca proprietar utilizatorul SYS.
Instantaneul este o copie locala a unei baze de date situate la distanta si este utilizat in contextul bazelor de date distribuite
Exista doua tipuri de instantanee :**simple si complexe**
Un instantaneu simplu este un instaneu a cărui introgare de definire este realizata astfel incât în fiecare rand din instantaneu corespunde cu un rand sau o parte de rand din tabelul master,care in mod practic nu trebuie să conţina :clauza GROUP BY, CONNECT BY, operatorul DISTINCT, UNION, INTERSECT , MINUS
Prin urmare in momentul crearii unui instantaneu se vor crea urmatoarele obiecte :

- In baza de date situata la distanta, un tabel de baza denumit **$SNAP_nume_instantaneu**, care va contine rezultatele interogarii ;
- Pentru cheia primara a acestui tabel va fi creat un index ;
- Pentru fiecare instantaneu, Oracle creaza o vedere ce va avea acelasi nume ca şi instantaneul

Reimprospatarea instantaneelor
Reimprospatarea unui instantaneu se poate face in doua moduri :
1.*manual*- folosind comandana *DBMS_SNAPSHOT.REFRESH()*
2.*automat* – prin specificarea modului si a intervalului la care are loc improspatarea in momentul crearii
Crearea instantaneelor[10]
CREATE SNAPSHOT nume_instantaneu
[REFRESH[FAST|COMPLETE|FORCE][START WITH data][NEXT data]
[WITH PRIMARY KEY|WITH ROWID]]
[FOR UPDATE]
AS subinterogare
Exemplu :Să se creeze un instantaneu **titluri_ro** care să contina toate inregistrarile din tabela titluri aflata in schema utilizatorului scott din baza de date Biblio, reimprospatarea va fi rapida si care va avea loc la o ora dup ace instantaneul a fost creat, urmatoarele reimprospatarii facandu-se la interval de 7 zile.

CREATE SNAPSHOT titluri_ro
REFRESH FAST
START WITH sysdate+1/24
NEXT sysdate+7
AS
*SELECT * FROM scott.titluri@biblio;*
Modificarea si distrugerea instantaneelor
Modificare se face folosind comanda ALTER SNAPSHOT
EXE:ALTER SNAPSHOT titluri_ro REFRESH COMPLETE NEXT sysdate+1;
Distrugerea se face utlizand comanda DROP SNAPSHOT;

Legaturi la baza de date BIBLIOTECA[10]

O legatura la o baza de date este creata pentru a facilita conectarea la o baza de date situata la distanta, atunci cand se foloseste un nume global, aceasta legatura defineste calea catre baza de date situata la distanta.
Fiecae baza de date continuta intr-o baza de date distribuita are propriul sau nume , numit nume global al bazei de date, format din prefixarea numelui de domeniu al retelei pe care sa afla baza de date cu numele individual al bazei de date.
Numele de domeniu si numele bazei de date este dat de parametrul de initializare DB_DOMAIN,respectiv DB_NAME
Exemplu daca domeniile de retea **romania.ro si europa.ro** contin fiecare o baza de date cu denumrea biblioteca, atunci aceste baze de date vor avea numele global **biblioteca.romania.ro (biblioteca.europa.ro)**
Pentru a facilita cererile unei aplicatii ce ruleaza pe o baza de date distribuita (BDD), Oracle utilizeaza legaturile unei baze de date.
O legatura a unei baze de date este un obiect al bazei de date locale care permite accesul la obicte dintr-o baza de date la distanta.
Exista trei tipuri de legaturi la baze de date :

1. private-legatura este creata pentru un anumit utilizator şi nu poate fi folosita decăt de utilizatorul specificat
2. public-legatura este creata pentru grupul de utilizatori PUBLIC şi poate fi folosita de catre toti utilizatori BD
3. global-legatura este gestionata automat de catre un serviciu de reţea

Crearea legaturilor bazei de date

Pentru a crea o legatura se utilizeaza comanda **CREATE DATABASE LINK**
Forma generala este :
CREATE [PUBLIC]DATABASE LINK nume_legatura
[CONNECT TO user IDENTIFIED BY parola]

Exemplu
Sa se creeze o legatura private la baza de date **biblioteca cu numele biblio si parola biblio**
CREATE DATABASE LINK BIBLIOTECA.ROMANIA.RO
CONNECT TO BIBLIO IDENTIFIED BY BIBLIO

Odata create , utilizatori pot accesa datele bazei de date respective prin folosirea numelor globale ale obiectelor.
De exemplu SELECT * FROM EDITURI@BIBLIOTECA.ROMANIA.RO;
O legatura globala se creeaza cu ajutorul produsului Oracle Name Server
Observaţie :Legaturile nu se pot crea decat daca utilizatorii detine privilegiul de sistem CREATE DATABASE LINK, sau CREATE PUBLIC DATABAE LINK ,iar in baza de date situata la distanta privilegiul CREATE SESSION,Produsul NET9 trebuie instalat atat in nodul local căt şi in nodul situat la distanta

Stergerea legaturilor bazei de date

O legatura la o baza de date situate la distanta se poate sterge din baza de date locala cu ajutorul comenzii:
DROP DATABASE LINNK nume_legatura ;

Limitarile bazelor de date distribuite
Intr-o baza de date distribuita ,se poate stabili cateva restrictii pentru a imbunatati perfomantele :

- limitarea numarului de tranzactii distribuite per nod,parametrul distributed_transaction controleaza numarul de tranzactii distribuite,in care o anumita instanta poate participa in mod concurent
- să se stabileasca taria punctelor de finalizare pentru fiecare instanta

Procese distribuite
Un proces distribuit se caracterizeaza prin utilizarea a mai mult de un singur procesor pentru a diviza procesarea unei sarcini individuale.
Un exemplu este configuratia client-server.
Cele mai importante avantaje sunt :

- Responsabilitatea pentru procesarea datelor revine sistemului server
- Aplicatiile client pot fi concepute fara a lua in calcul localizarea datelor
- Aplicatia poate beneficia de caracteristicile de procesare multitasking si de partajare a memoriei ale serverului Oracle

Secvente
O secventa situata la distanta poate fi referită daca urmatoarele doua propozitii sunt adevarate :

- toate secventele, tabele actualizate, tabele selectate sunt situate în acelasi nod

Exemplu
INSERT INTO LISTA_INREG@INVENTAR_01
SELECT SECVENTA_1.NEXTVAL@INVENTAR_01,COD,NUME,PRET
FROM LIVRARI@INVENTAR_01

Integritate prin referinta
Oracle nu permite declararea integritatii prin referinta intre noduri distincte ale unei baze de date distribuite.Nu se poate defini pe un table o restrictie de integritate prin specificarea unei chei straine care sa faca referire la cheia principala sitiata la distanta, insa relatiile părinte/fiu sunt premise intre bazele de date distribuite, aceasta poate fi mentinuta numai prin folosirea declansatorilor.

Interogari distribuite
Se poate crea o procedura rezidenta sau un declansator pentru o interogare distribuita.
In nodul local, Oracle descompune interogarea in interogarile la distanta corespunzatoare,pe care le transmite nodurilor situate la diastanta, in vederea executiei.
In continuare , fiecare nod situat la distanta proceseaza interogarea şi returneaza bazei de date locale rezultatele interogareii.Nodul local efectueaza post-procesările necesare şi returnează un răspuns aplicaţiei care a initiat interogarea.

9.4.8 Crearea meniului principal BIBLIOTECA

Procesul de creare a unui meniu pentru intreaga aplicaţiei poate fi structurat în cinci etape:

1. planificarea si proiectarea meniului
2. crearea efectiva a meniurilor si submeniurilor,adica declarea , folosind Menu Designer

3. specificarea actiunilor ce vor fi declansate prin selectarea optiunilor meniului
4. generarea programului , a secventei de comenzi DEFINE PAD,POPUP,BAR,ON SELECTION BAR)
5. rularea si testarea programului generat anterior

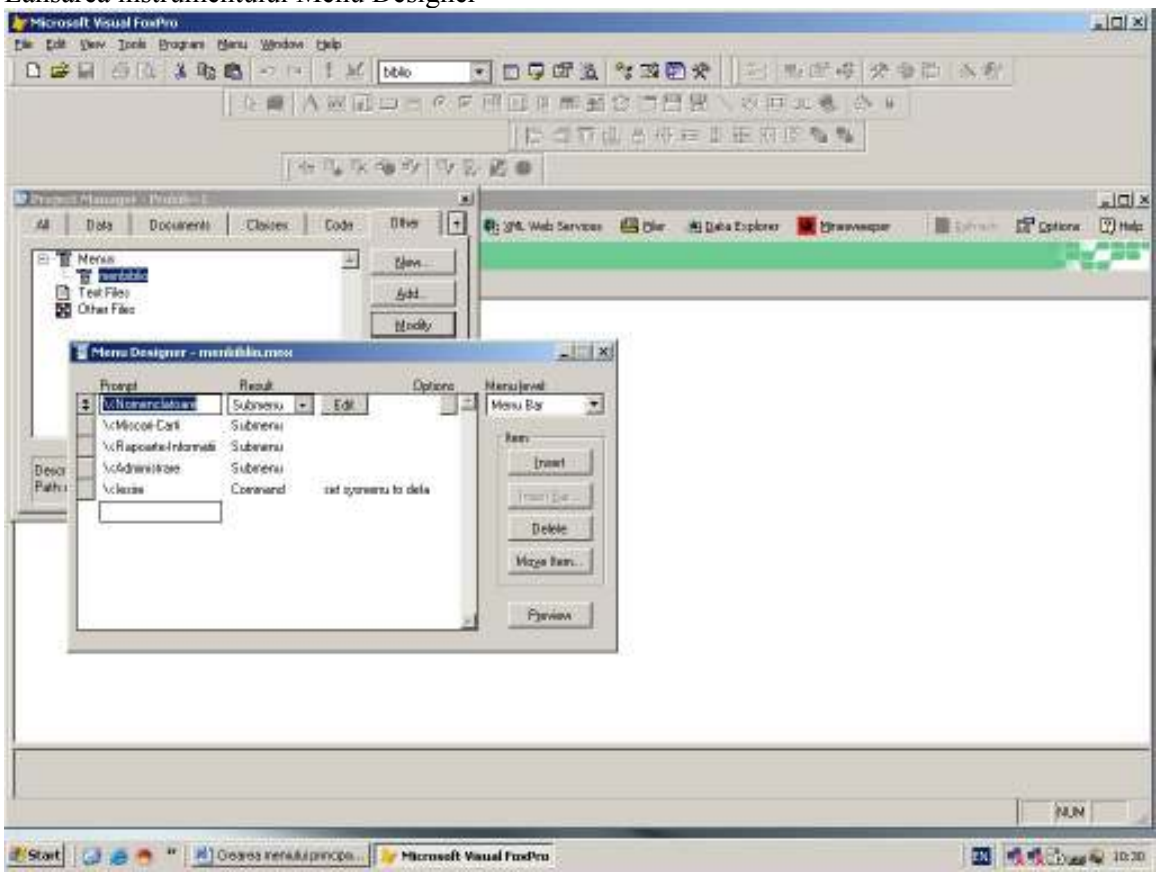
1.planificarea si proiectarea meniului principal al aplicatiei Biblioteca

- **meniul-bara** va fi compus din urmatoarele submeniuri:
  - **Nomenclatoare**-ce actualizeaza datele din tabelele de referinta cum sunt :Autori, Clasificarea zecimala universala (CZU),Edituri,preluarea intiala a cartilor din biblioteca (inventar)
  - **Miscari-Carti-**care va permite accesul la formularele ce actualizeaza datele din tabelele :comenzi de carti, titluri,titluri_autori, titluri_cuvinte, exemplare, clienti , Imprumut, reviste, ziare
  - **Rapoarte-informatii**:va cuprinde o serie de rapoarte privind registrul inventar, registrul de miscari intrari/iesiri, registrul cititorilor inscrisi, imprumutul cartilor,fisa de catalog in forma alfabetica si sistematica
  - **Administrarea bazei** de date si configurare care permite intretinerea bazei de date (crearea copiilor de siguranta/arhivarea si refacerea BD,replcarea bazei de date distribuite,configurarea mediului software/hardeware in care rezida aplicatia (retea,imprimanta,configurarea client-server cu baza de date ORACLE)
  - **Asistenta** care va constituie un help al aplicatiei
  - **Iesire**

Schema de baza a meniului principal al aplicatiei BIBLIOTECA

Nomenclatoare	**Mişcari-carţi**	**Rapoarte-info.**	**Administrare BD**	**Asistenta/iesire**
Autori	Comenzi	Registrul intrari	Crearea copiilor	Continut si index
Clasificare zec. Universala	Preluarea initiala a cartilor(inventar)	Registrul inventar	Arhivarea si refacerea	Iesire
Edituri	Intrari de carti	Fişa carti	Replicarea bd	
Vizualizare autori	Autori de titluri	Fisa de imprumut	Configurare Imprimanta	
Vizualizare czu	Cuvinte cheie	Catalog alfabetic	Configurare retea	
	Cote	Catalog sistematic		
		Situatia cartilor nesrestituite		
Vizualizare edituri	Clienti	Situatia cartilor		
	Reviste			
	Ziare			
	Imprumut			

2.crearea/definirea meniului principal si submeniurilor corespunzatoare
Lansarea instrumentului Menu Designer

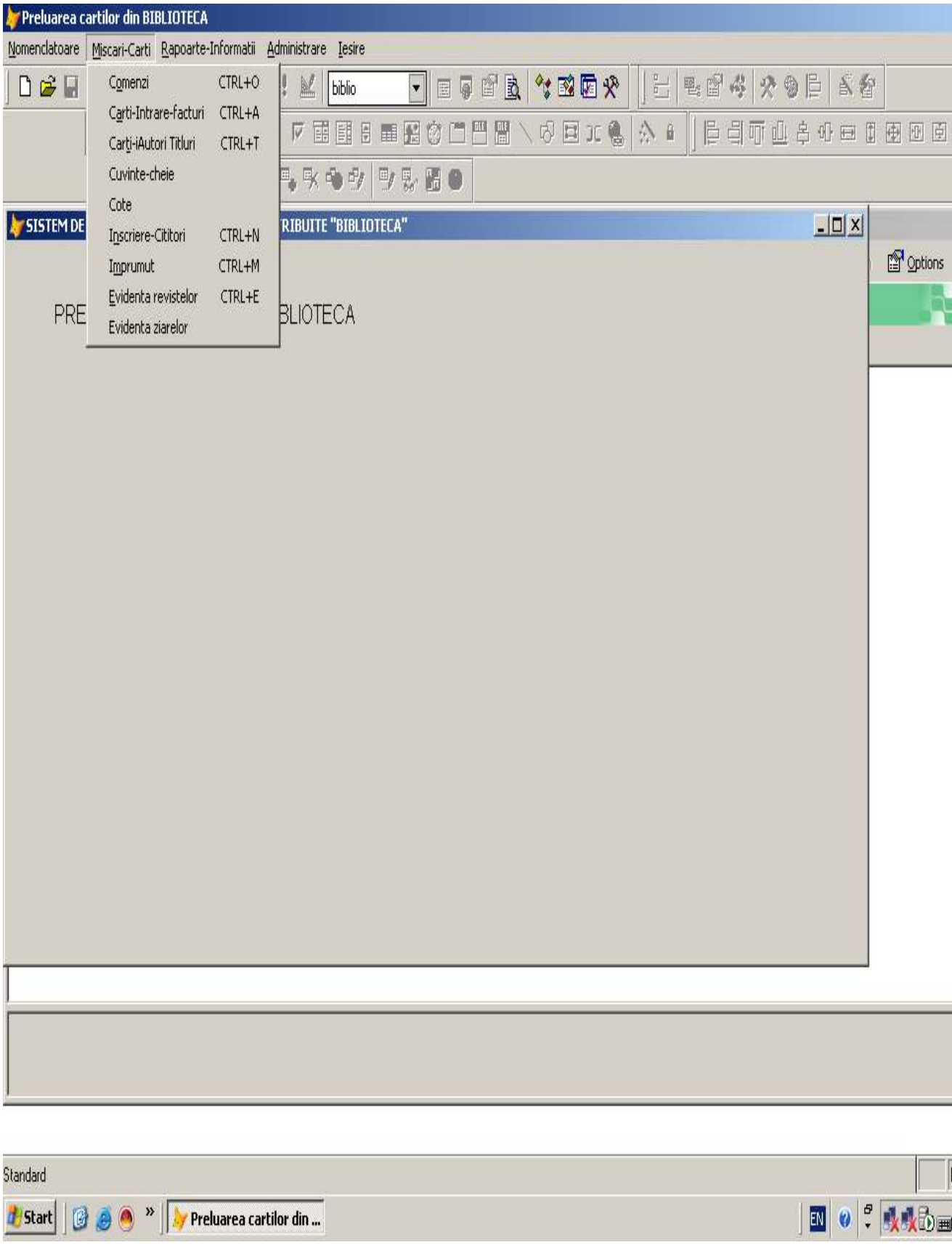


3 Specificatii pentru meniul principal al aplicatiei BIBLIOTECA

Titlu optiune sau submeniu	Nume program	
Meniul-bara	Submeniu	
Nomenclatoare	Submeniu	
Autori	Command	Do form frmautori
Clasificare zecimala universala(CZU)	Command	Do form frmczu
Edituri	command	Do form frmedit
Vizualizare autori	Command	Report form rep_autori preview
Vizualizare Czu	Command	Report form rep_czu preview
Vizualizare edituri	Command	Report form rep_edituri preview
Miscari-Carti	Submeniu	
Intrari de carti	Command	Do form frmtitlu

Autori-carti	Command	Do form frmtitluri_autori
Exemplare (Cote)	command	Do form frmcote
Cuvinte-cheie	Command	Do form frmtitluri_cuvinte
Clienti(Cititori)	Command	Do form frmclienti
Imprumut	Command	Do form frmimprumut
Reviste	Command	Do form frmreviste
Ziare	Command	Do form frmziare
Rapoarte-informatii	Submeniu	
Registrul de intrari	Command	Report form rep_rgintrari
Registrul inventar	Command	Report form rep_inventar preview
Fisa carti	Command	Do form frmcarti
Fisa de contract(imprumut)	Command	Do form frmimp
Fisa catalog	Command	Do form frmcatalog
Situatia cartilor nerestituite	Command	Do situatie_nerestituit
Administrarea bazei de date/configurare	Submeniu	
Administrarea BD	Submeniu	
Crearea copiilor de siguranta	Command	Do setup
Refacerea BD	Ccommand	Do refacere_bd
Replicarea BD	Command	Do replicarea_bd
Configurare retea	Command	Do config_retea
Configurare imprimanta	Submeniu	
Imprimanta	Bar #	_mfi_sysprint
Pagina	Bar #	_mfi_pgset
Asistenta	Submeniu	
Continutu si index	Command	Help
Iesire	Command	Set sysmenu to default

9.4.9 Anexe programe realizate in Visual Foxpro cu serverul Oracle 9i



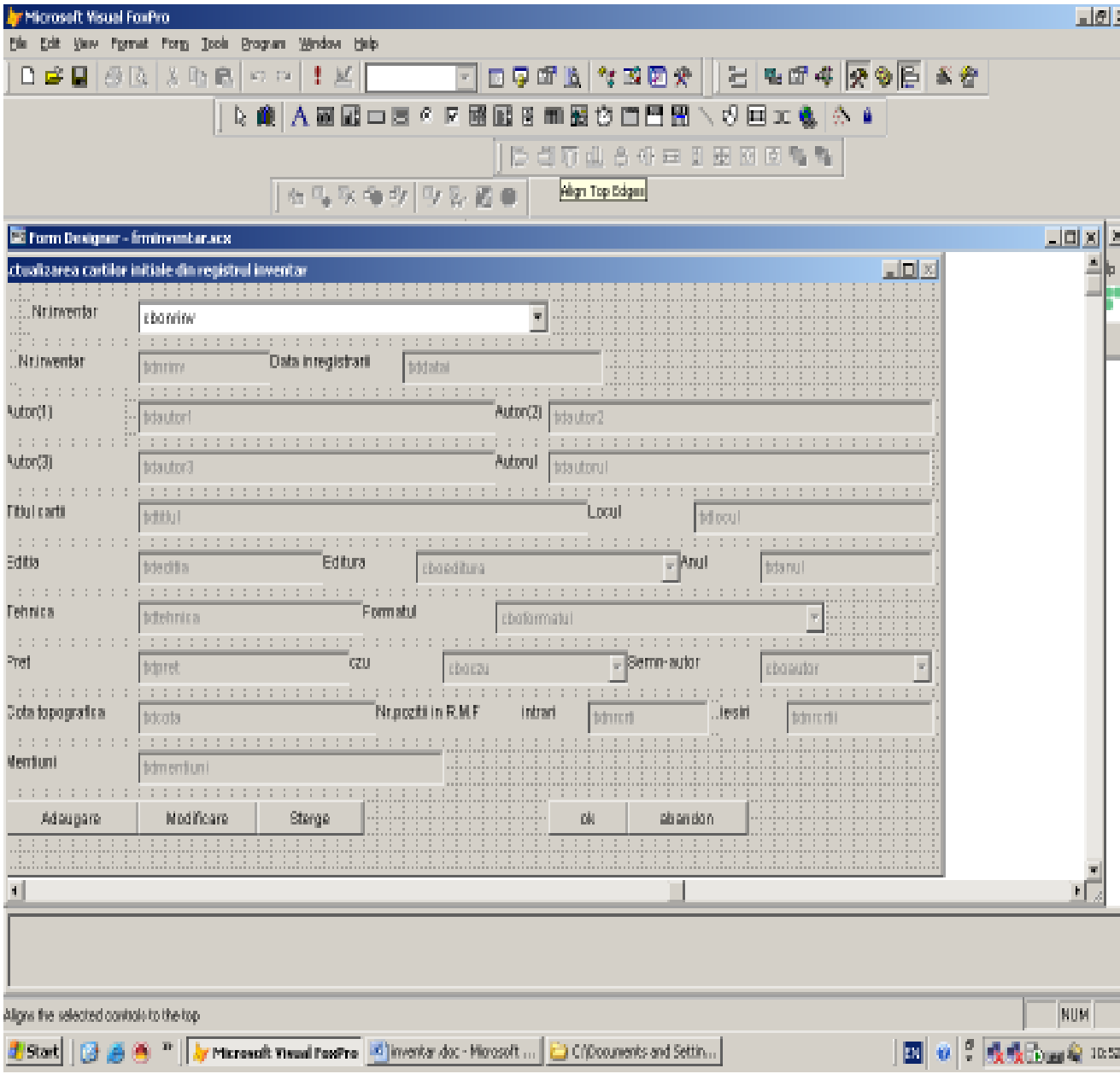
Listing 1 program pentru preluarea cartilor din BIBLIOTECA (PRINCIPAL)

```
set safety off
set century on
set exact on
set date to british
on shutdown do prgies
do form c:\biblio\forms\frmlog
x=rat("\",sys(16))
zona_de_lucru="'"+left(sys(16),x-1)+"'"

set defa to &zona_de_lucru
set path to database,imagini,forms,meniuri,help,prog,reports
open database c:\biblio\database\biblio shared
_screen.icon='green.ico'
_screen.picture='background.jpg'
_screen.windoWstate=2
_screen.caption='Preluarea cartilor din BIBLIOTECA'
do c:\biblio\meniuri\menbiblio.mpr


read events
```

LISTING 2 PENTRU PROGRAMUL PRGIES

```
close data all

on key
do compact
clear events
quit
```

Anexa 1 Formular pentru preluarea initiala a cartilor dintr-o BIBILOTECA




```
*********************
LISTING 1 INIT
*********************

PUBLIC verr(7),a,nrinv_,grupa_,dendiviz_,editura_
iF !USED('vinventar')
USE vinventar IN 0 EXCLUSIVE
ENDIF
IF !USED('vczu')
```

```
USE vczu IN 0 EXCLUSIVE
endif
if!USED('vautori')
USE vautori IN 0 EXCLUSIVE
ENDIF
IF !USED('vedituri')
USE vedituri IN 0 EXCLUSIVE
ENDIF

SELECT vinventar
REQUERY()
IF EOF() OR BOF()
GO top
ENDIF
thisform.cbonrinv_refresh
thisform.dezactivare
```

LISTING 2 metoda activare a formularului frminventar

```
thisform.cbonrinv.Enabled=.f.
thisform.txtnrinv.Enabled=.t.
thisform.txtdatai.Enabled=.t.
thisform.txtautor1.Enabled=.t.
thisform.txtautor2.Enabled=.t.
thisform.txtautor3.Enabled=.t.
thisform.txtautorul.Enabled=.t.
thisform.txttitlul.Enabled=.t.
thisform.txtlocul.Enabled=.t.
thisform.txteditia.Enabled=.t.
thisform.cboeditura.Enabled=.t.
thisform.txtanul.Enabled=.t.
thisform.txttehnica.Enabled=.t.
thisform.cboformatul.Enabled=.t.
thisform.txtpret.Enabled=.t.
thisform.cboczu.Enabled=.t.
thisform.cboautor.Enabled=.t.
thisform.txtcota.Enabled=.t.
thisform.txtnrcrti.Enabled=.t.
thisform.txtnrcrtii.Enabled=.t.
thisform.txtmentiuni.Enabled=.t.
thisform.cmdadd.enabled=.f.
thisform.cmdmod.Enabled=.f.
thisform.cmdstg.Enabled=.f.
```

listing 3 cbonrinv_refresh

```
*********************
FOR i=1 TO thisform.cbonrinv.ListCount
```

```
IF
ALLTRIM(thisform.cbonrinv.ListItem(i))==ALLTRIM(STR
(vinventar.nrinv))
thisform.cbonrinv.ListIndex=i
exit
ENDIF
ENDFOR
```

Listing 4 dezactivare

```
**********************
thisform.txtnrinv.Enabled=.f.
thisform.txtdatai.Enabled=.f.
thisform.txtautor1.Enabled=.f.
thisform.txtautor2.Enabled=.f.
thisform.txtautor3.Enabled=.f.
thisform.txtautorul.Enabled=.f.
thisform.txttitlul.Enabled=.f.
thisform.txtlocul.Enabled=.f.
thisform.txteditia.Enabled=.f.
thisform.cboeditura.Enabled=.f.
thisform.txtanul.Enabled=.f.
thisform.txttehnica.enabled=.f.
thisform.cboformatul.Enabled=.f.
thisform.txtpret.Enabled=.f.
thisform.cboczu.Enabled=.f.
thisform.cboautor.Enabled=.f.
thisform.txtcota.Enabled=.f.
thisform.txtnrcrti.Enabled=.f.
thisform.txtnrcrtii.Enabled=.f.
thisform.txtmentiuni.Enabled=.f.
thisform.cmdadd.enabled=.t.
thisform.cmdmod.Enabled=.t.
thisform.cmdstg.Enabled=.t.
```

listing 5 scrie/buffer

```
**********************
SELECT vinventar

RELEASE ALL LIKE sir*
a=TABLEUPDATE()
IF !a
=AERROR(verr)
FOR i=3 TO 7
    ii=STR(i,1)
    sir&ii=NVL(verr(i),"NULL")
    sir_=sir&ii
    IF TYPE("sir_")="N"
```

```
       sir&ii=STR(sir&ii,10)
    ENDIF
    ENDFOR
    INSERT INTO temp values(1,STR(verr(1),7))
    INSERT INTO temp values(2,verr(2))
    INSERT INTO temp values(3,sir3)
    INSERT INTO temp values(4,sir4)
    INSERT INTO temp values(5,sir5)
    INSERT INTO temp values(6,sir6)
    INSERT INTO temp values(7,sir7)

    DO case

    CASE verr(1)=1585
    MESSAGEBOX('inregistrarea actualizata a fost
modificata de altcineva')
    CASE verr(1)=1526 AND verr(5)=1 AND
"pk_nrinv"$verr(2)
    MESSAGEBOX('se repeta valoarea atributului cota
!')
    *CASE verr(1)=1526 AND verr(5)=2290 AND
"ck_titlu"$verr(2)
    *MESSAGEBOX('valoarea atributului titlu
depaseste limita stabilita !')
    CASE verr(1)=1526 AND verr(5)=1400 AND
"pk_nrinv"$verr(2)
    MESSAGEBOX('valoarea nula nu se poate introduce
')
    CASE verr(1)=1526 AND verr(5)=20015
    MESSAGEBOX(SUBSTR(verr(3),31,65))
    OTHERWISE
    MESSAGEBOX('Eroare netrata !')

    ENDCASE
    =TABLEREVERT()
    ENDIF
```

Listing 6 programe/referitoare la Butonul adauga un articol
Pentru metoda click a formularului frminventar

```
**************************************************
*********

IF !USED('vinventar')
USE vinventar IN 0 EXCLUSIVE
endif
SELECT vinventar
nrinv_=vinventar.nrinv
grupa_=vautori.grupa
```

```
dendiviz_=vczu.dendiviz
editura_=vedituri.editura

APPEND BLANK
replace nrinv  WITH nrinv_,datai WITH DATE(),semn
WITH 'GRUPA',czu WITH 'CZU'

thisform.scrie_buffer

REQUERY()

GO bottom
thisform.cbonrinv_refresh

thisform.Refresh

thisform.activare

listing 7-metoda modificare
thisform.activare

listing 8 -butonul stergere articole - metoda click
**************************************************
*
SELECT vinventar
DELETE
thisform.scrie_buffer
IF a
IF EOF()
GO bottom
ELSE
IF BOF()
GO top
ENDIF
ENDIF
b_=recno()
REQUERY()
thisform.cbonrinv.Requery
GO MIN(b_,RECCOUNT())
thisform.cbonrinv_refresh
thisform.Refresh
ENDIF

Listing 9 butonul OK
thisform.scrie_buffer
IF !a
    thisform.txtnrinv.SetFocus
    ELSE
    b_=RECNO()
```

```
    SELECT vinventar
    REQUERY()
    thisform.cbonrinv.Requery
    GO b_
    thisform.cbonrinv_refresh
    thisform.Refresh
    thisform.dezactivare
    ENDIF
```

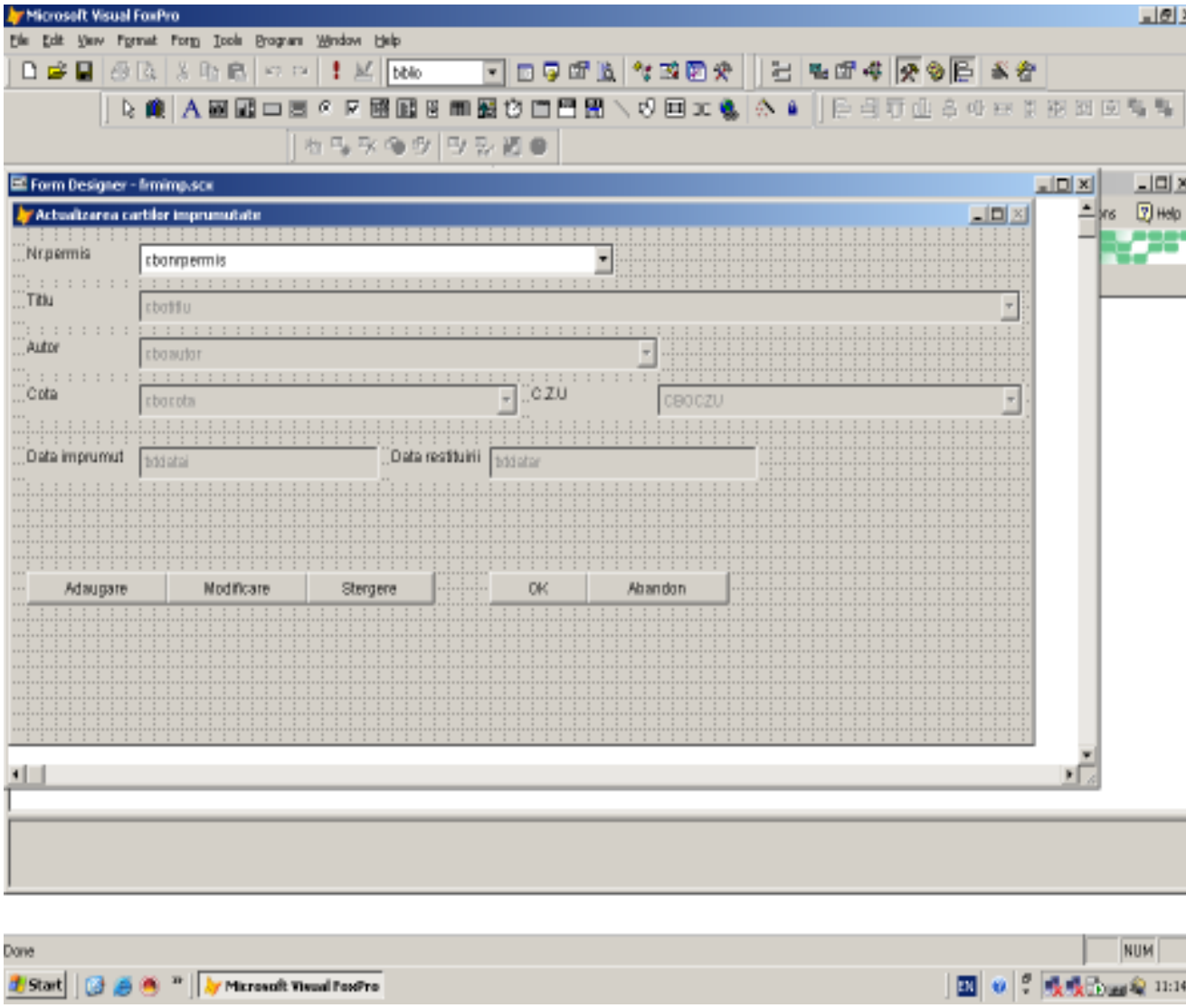
Listing 10 – Butonul Abandon

```
=TABLEREVERT(.t.)
thisform.Release
```

program pentru actualizarea tabelei derivate tabderiv_inventar.prg

```
**crearea tabelei derivate vautori
IF NOT DBUSED('biblio')
    OPEN DATABASE "c:\biblio\database\biblio" SHARED
endif
#include foxpro.h
CREATE SQL  VIEW vinventar  CONNECTION biblio as select *
from inventar
=DBSETPROP('vinventar','view','tables','inventar')
=DBSETPROP('vinventar.nrinv','field','keyfield',.t.)
=DBSETPROP('vinventar.nrinv','field','updatable',.T.)
=DBSETPROP('vinventar.datai','field','updatable',.T.)
=DBSETPROP('vinventar.autor1','field','updatable',.t.)
=DBSETPROP('vinventar.autor2','field','updatable',.t.)
=DBSETPROP('vinventar.autor3','field','updatable',.t.)
=DBSETPROP('vinventar.autorul','field','updatable',.t.)
=DBSETPROP('vinventar.titlul','field','updatable',.t.)
=DBSETPROP('vinventar.locul','field','updatable',.t.)
=DBSETPROP('vinventar.editia','field','updatable',.t.)
=DBSETPROP('vinventar.editura','field','updatable',.t.)
=DBSETPROP('vinventar.anul','field','updatable',.t.)
=DBSETPROP('vinventar.tehnica','field','updatable',.t.)
=DBSETPROP('vinventar.formatul','field','updatable',.t.)
=DBSETPROP('vinventar.pret','field','updatable',.t.)
=DBSETPROP('vinventar.cota','field','updatable',.t.)
=DBSETPROP('vinventar.czu','field','updatable',.t.)
=DBSETPROP('vinventar.semn','field','updatable',.t.)
=DBSETPROP('vinventar.nrcrti','field','updatable',.t.)
=DBSETPROP('vinventar.nrcrtii','field','updatable',.t.)
=DBSETPROP('vinventar.mentiuni','field','updatable',.t.)
=DBSETPROP('vinventar','view','updatetype',db_update)
=DBSETPROP('vinventar','view','wheretype',DB_KEYANDMODIFIED)
=DBSETPROP('vinventar','view','sendupdates',.T.)
```

Anexa 2 –Formular pentru actualizarea cartilor imprumutate frmimprumut



Listing 1 metoda init a formularului frmimprumut

```
PUBLIC
verr(7),a,idisbn_,nrpermis_,nume_,pren_,isbn_,titlu_,nrinv_,a
utor_,cota_,dendiviz_,clasa_
iF !USED('vimprumut')
USE vimprumut IN 0 EXCLUSIVE
ENDIF
IF !USED('vtitluri')
USE vtitluri IN 0 EXCLUSIVE
ENDIF
if!USED('vclienti')
use vclienti in 0 exclu
ENDIF
iF !USED('vexemplare')
USE vexemplare IN 0 EXCLUSIVE
ENDIF

if!USED('vtitluri_autori')
```

```
use vtitluri_autori in 0 exclu
ENDIF
IF !USED('vczu')
USE vczu IN 0 EXCLUSIVE
ENDIF

SELECT vimprumut
REQUERY()
IF EOF() OR BOF()
GO top
ENDIF
thisform.cbonrpermis_refresh
thisform.dezactivare
```

listing 2 –metoda activare a formularului frmimprumut

```
thisform.cbonrpermis.Enabled=.f.
thisform.cbotitlu.Enabled=.t.
thisform.cboautor.Enabled=.t.
thisform.cbocota.Enabled=.t.
thisform.cboczu.Enabled=.t.
thisform.txtdatai.Enabled=.t.
thisform.txtdatar.Enabled=.t.
thisform.cmdadd.enabled=.f.
thisform.cmdmod.enabled=.f.
thisform.cmdstg.enabled=.f.
```

listing 3 – pentru adaugarea de inregistrari in tabela IMPRUMUT aflata pe serverul bazei de date ORACLE9i

```
SELECT vimprumut

idisbn_=vtitluri.idisbn
isbn_=vtitluri.isbn
titlu_=vtitluri.titlu
nrinv_=vtitluri.nrinv

nrpermis_=vclienti.nrpermis
nume_=vclienti.nume
pren_=vclienti.pren

autor_=vtitluri_autori.autor

cota_=vexemplare.cota

dendiviz_=vczu.dendiviz
clasa_=vczu.clasa


APPEND BLANK
replace nrpermis WITH nrpermis_,idisbn WITH idisbn_,isbn WITH
isbn_,nrinv WITH nrinv_,dataimp WITH DATE(),;
```

```
autor WITH autor_,titlu WITH titlu_,czu WITH
dendiviz_+clasa_,cota WITH cota_,numepren WITH
nume_+pren_,datarest WITH DATE()

thisform.scrie_buffer

REQUERY()

GO bottom

thisform.cbonrpermis_refresh

thisform.Refresh

thisform.activare
```

listing 4 pentru stergerea de articole atat din tabela derivate vimpumut cat si din serverul BD IMPRUMUT aflata in ORACLE

```
SELECT vimprumut
DELETE
thisform.scrie_buffer
IF a
IF EOF()
GO bottom
ELSE
IF BOF()
GO top
ENDIF
ENDIF
b_=recno()
REQUERY()
thisform.cbonrpermis.Requery
GO MIN(b_,RECCOUNT())
thisform.cbonrpermis_refresh
thisform.Refresh
ENDIF
```

LISTING 5 PENTRU ACCEPTAREA BUTONULUI OK

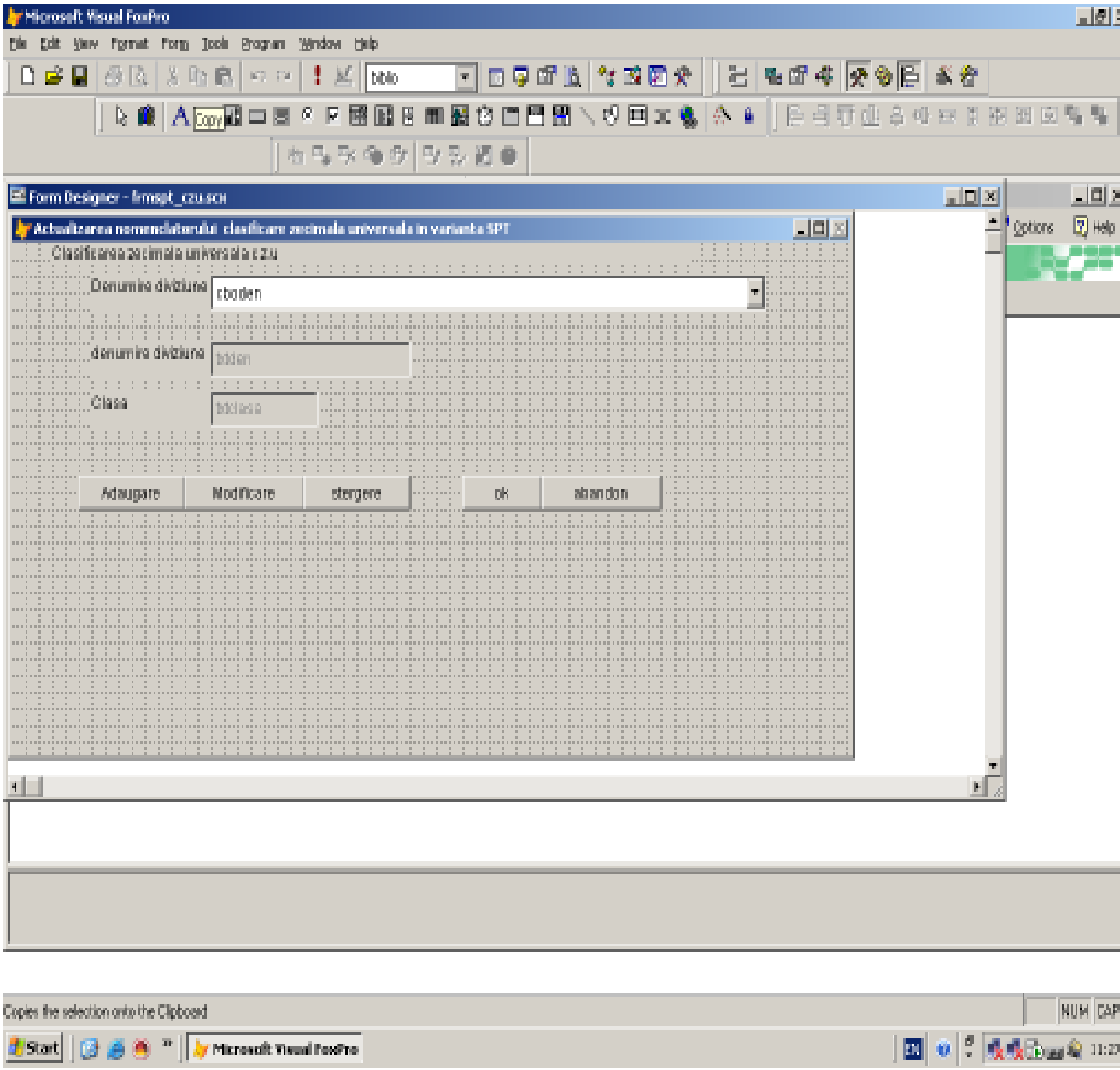
```
thisform.scrie_buffer
IF !a
    thisform.txtdatai.SetFocus
    ELSE
    b_=RECNO()
    SELECT vimprumut
    REQUERY()
    thisform.cbonrpermis.Requery
    GO b_
    thisform.cbonrpermis_refresh
    thisform.Refresh
    thisform.dezactivare
    ENDIF
```

LISTING 6 –ACTUALIZAREA TABELEI DERIVATE VIMPRUMUT

```
*crearea tabelei derivate vedituri
```

```
IF NOT DBUSED('biblio')
    OPEN DATABASE "c:\biblio\database\biblio" SHARED
endif
#include foxpro.h
CREATE SQL  VIEW vimprumut  CONNECTION biblio as select *
from imprumut
=DBSETPROP('vimprumut','view','tables','imprumut')
=DBSETPROP('vimprumut.nrpermis','field','keyfield',.T.)
=DBSETPROP('vimprumut.idisbn','field','keyfield',.t.)
=DBSETPROP('vimprumut.nrpermis','field','updatable',.T.)
=DBSETPROP('vimprumut.idisbn','field','updatable',.T.)
=DBSETPROP('vimprumut.nrinv','field','updatable',.T.)
=DBSETPROP('vimprumut.dataimp','field','updatable',.T.)
=DBSETPROP('vimprumut.autor','field','updatable',.T.)
=DBSETPROP('vimprumut.titlu','field','updatable',.T.)
=DBSETPROP('vimprumut.czu','field','updatable',.T.)
=DBSETPROP('vimprumut.cota','field','updatable',.t.)
=DBSETPROP('vimprumut.numepren','field','updatable',.t.)
=DBSETPROP('vimprumut.datarest','field','updatable',.t.)
=DBSETPROP('vimprumut.isbn','field','updatable',.t.)
=DBSETPROP('vimprumut','view','updatetype',DB_UPDATE)
=DBSETPROP('vimprumut','view','wheretype',DB_KEYANDMODIFIED)
=DBSETPROP('vimprumut','view','sendupdates',.T.)
```

ANEXA-3 FORMULAR PENTRU ACTUALIZAREA TABELEI CZU FOLOSIND TEHNOLOGIA SPT



LISTING 1 Metoda Load a formularului frmspt_czu

```
public
verr(7),a,dendiviz_,clasa_,NRCONEXIUNE,EXISTA_TAG,idden_
IF NOT DBUSED('biblio')
  OPEN DATABASE "\biblio\database\biblio " shared
endif

IF !USED('vczu')
   USE vczu IN 0 EXCLUSIVE
ENDIF

SELECT vczu

FOR i=1 TO TAGCOUNT()
IF ALLTRIM(UPPER(i))=idden
exista_tag=.t.
```

```
ENDIF
ENDFOR
IF !exista_tag
INDEX on idden TAG idden
ENDIF



SELECT vczu
nrconexiune=CURSORGETPROP('connecthandle')
vsuces=SQLEXEC(nrconexiune,"select dendiviz,clasa from czu
order by dendiviz","crsczu")
IF vsuces<0
MESSAGEBOX('eroare !')
RETURN .f.
ENDIF
```

Listing 2-metoda Destroy a formularului

```
IF TXNLEVEL()>0
   IF MESSAGEBOX('salvati ultimele
modificari?',32+4,'ultimele nu au fost validate')=6
   END TRANSACTION
   ELSE
   ROLLBACK
   ENDIF
   ENDIF

   IF thisform.asters=.t.
   SELECT czu
   thisform.scrie_buffer
   ENDIF
```

Listing 3-cmdadd -adaugarea de intergistrari in tabela vczu aflata pe clientul 1……..n si tabela czu aflata pe serverul ORACLE9I

```
BEGIN TRANSACTION
INSERT INTO vczu values(0,'','')

thisform.scrie_buffer
thisform.activare
thisform.txtden.SetFocus
```

listing 4-cmdmod –pentru modificarea articolelor din baza de date

```
BEGIN TRANSACTION
thisform.activare
thisform.txtden.SetFocus
```

listing 5 cmdstg-stergerea de articole

```
SELECT vczu
IF MESSAGEBOX('sunteti sigur ?',32+4,'clasa va fi stearsa
!!!')=6
    DELETE
    thisform.asters=.t.

thisform.scrie_buffer
```

```
ENDIF
```

Listing 6-cmdok –pentru acceptarea datelor introduce

```
END TRANSACTION
SELECT vczu
thisform.scrie_buffer
thisform.Refresh
thisform.dezactivare
thisform.cboden.Requery
thisform.cbogrupa_refresh
```

listing 7-pentru actualizarea cursorului

```
# include foxpro.h

nrconexiune=SQLCONNECT('biblio')
IF nrconexiune<1
=MESSAGEBOX("conexiune esutata",0,"abandon")
RETURN
ENDIF

VSuces=SQLEXEC(nrconexiune,'select * from czu ','cczu')
IF vsuces<1
=MESSAGEBOX("interogare fata rezultat",0,"rezultatul
interogarii")
RETURN
ENDIF


SET MULTILOCKS ON
=CURSORSETPROP("buffering",3,"cczu")
=CURSORSETPROP('tables','czu','cczu')
=CURSORSETPROP('keyfieldlist','idden','cczu')
=CURSORSETPROP('updatablefieldlist','idden,dendiviz,clasa','c
czu')
=CURSORSETPROP('updatenamelist','idden czu.idden,dendiviz
czu.dendiviz,clasa czu.clasa','cczu')
=CURSORSETPROP('wheretype',DB_KEY,'cczu')
=CURSORSETPROP('sendupdates',.t.,'cczu')
=CURSORSETPROP('updatetype',DB_UPDATE,'CCZU')
RETURN
```

Listing 8-Pentru testarea conexiunii

```
IF !DBUSED('biblio')
OPEN DATABASE "\biblio\database\biblio" SHARED
endif
NRCONEXIUNE=SQLCONNECT('biblio')
IF nrconexiune<1
=MESSAGEBOX("conexiune esuata",0,"abandon")
RETURN
ENDIF
vsuuces=SQLEXEC(nrconexiune,'select * from edituri
','cedituri')
IF vsuuces>0
SELECT cedituri
```

```
BROWSE
ELSE
=MESSAGEBOX('interogare fara rezultat",0,"rezultatul
interogarii")
ENDIF
RETURN
```

9.4.10 ANEXE

ANEXA1

CUVINTE REZERVA TE

LIST ROWID
LOCK ROWNUM
ACCES DISTINCT LONG ROWS
ADD DOES MAJCEXTENTS RUN
ALL DROP MINUS SELECT
ALTER MODE SESSION
AND ELSE MODIFY SET
ANY ERASE MOVE SHARE
APPEND EVALUATE NEW SIZE
AS EXCLUSIVE NO AUDIT SMALLINT
ASC EXISTS NOCOMPRESS SPACE
ASSERT FILE NOLIST START
ASSIGN FLOAT NOSYSSORT SUCCESSFUL
AUDIT FOR NOT SYNONYM
BETWEEN FORMAT NOW AIT SYSDATE
BY FROM NULL SYS SORT
CHAR GRANT NUMBER TABLE
CHECK GRAPHIC OF TEMPORARY
CLUSTER GROUP OFFLINE THEN
COLUMN HAVING OLD TO
COMMENT IDENTIFIED ON TRIGGER
COMPRESS IF ONLINE UID
CONNECT IMAGE OPTIMIZE UNION
CONTAIN IMMEDIATE OPTION UNIQUE
CONTAINS IN OR UPDATE.
CRASH INCREMENT ORDER USER
CREATE INDEX PAGE USING
CURRENT INDEXED PARTITION VALIDATE
DATAPAGES INDEXPAGES PCTFREE VALUES
DATE INITIAL PRIOR VARCHAR
DBA INSERT PRIVILEGES VARGRAPHIC
DBLINK INTEGER PUBLIC VIEW
DECIMAL INTERSECT RAW WHENEVER
DEFAULT INTO RENAME WHERE
DEFINITION IS REPLACE WITH
DELETE LEVEL REPORT
DESC LIKE RESOURCE
REVOKE

ANEXA2

*OPERATORI UTILIZATI IN LIMBA JUL SQL*PLUS*

*A. Sintaxa operatorilor SQL*PLUS*

Operator	**Functie**
&	Specifics inlocuirile lexicale intr-un fisjer de comenzi executat cu START. Optiunile sunt inlocuite prin &s: prima prin &1, a doua prin
	Indica o variabila utilizator intr-o comanda SQL*PLUS. Se cere o valoare de fiecare data cand variabila & este gSsitS §i se cere o valoare cand se gSsete prima data variabila &&. Incheie o variabila de substitute urmatS de un caracter

B. Sintaxa operatorilor SQL

Operator	**Functie**
(...)	Include o subcerere continutS antr-o altS
	Delimiteaza o constants de tip caracter sau data calendaristicS. Un apostrof intr-o constants de tip caracter se preprezintS
	MarcheazS un nume de coloanS sau sinonim care confine caractere speciale. MarcheazS literalii intr-un format de tip datS
@	Precede un nume de legSturS la o bazS de date, intr-o clauzS FROM.

C. Operatori aritmetici SQL

Operator	**Functie**
+, -	Operatori unari + §i - (valori pozitive, respectiv valori negative)
***,/**	Operatori de inmultire §i impSrtire
+ -	Operatori de adunare §i scSdere
II	Concatenare de §iruri de caractere

D. Operatori logici

Operator	**Functie**
< > = > = < = ! = <>	Operatori de comparatie
NOT	Functia logica „NU"
AND	Functia logica „§1"
OR	Functia logica „SAU"
[NOT] IN(lista)	Egal cu oricare valoare din lista de valori
ANY	Indica o valoare oarecare dintr-o multime
ALL	Indica toate valorile unei multimi
[NOT] BETWEEN X AND Y	Valoarea unei variabile [nu] se gases.te in intervalul [x,yl
EXISTS	Conditia este adevarata daca o subcerere returneaza eel putin un rand
[NOT] LIKE	Compara valoarea unui camp cu un §ir de caractere. Daca in §irul de caractere urmeaza % se compara cu once secventa de caractere; - se compara cu once caracter;
IS [NOT] NULL	Operatorul specifics daca valoarea unei variabile este sau nu nula

E. Operatori utilizati in expresiile de cereri

Operator	**Functie**
UNION	Combina mai multe cereri, returnand reunirea liniilor selectate de cererile individuale
INTERSECT	Combina mai multe cereri, returnand intersectia liniilor selectate de cererile individuale
MINUS	Combina mai multe cereri, returnand diferenta liniilor selectate de doua cereri (randurile distincte selectate de prima cerere §i

F. Alti operatori SQL

Operator	**Functie**
(+)	Indica faptul ca o coloana ce il precede este coloana de jonctiune externa (OUTER JOIN
PRIOR	Definite relatiile parinte-fiu intre nodurile unei cereri structurate arborescent. Daca operatorul PRIOR precede o expresie din stanga unei egalitati, se face o selectie descendenta. Daca operatorul PRIOR precede o expresie din dreapta unei egalitati se va face o selectie

 ISSN 2069-7988
ISSN-L 2069-7988

ANEXA3

*PSEUDOCOLOANE UTILIZATE IN LIMBA JUL SQL*PLUS*

Numar coloana	**Valoare returnata**
LEVEL	1 pentru radacina, 2 pentru subdirector (copil) al radacinii etc. Se utilizeaza in comanda SELECT §i cluza CONNECT BY
NULL	Returneaza o valoare nula. Nu poate fi folosita in expresiile logice.
ROWID	Numarul de identificare al randului. Un identificator de rand este de tipul ROWID §i nu de tipul numar sau caracter.
ROWNUM	Numarul de ordine al randului selectat din tabela (numarand de la 1)
SYSDATE	Data calendaristica §i timpul curent
UID	Identificator de utilizator, este un numar unic pentru fiecare utilizator
USER	Returneaza numele utilizatorului curent

 ISSN 2069-7988
ISSN-L 2069-7988

CUPRINS

BIBLIOGRAFIE

1. Bâscă Octavian, Baze de date,Editura All, Bucureşti, 1997
2. Popescu Ileana, Letita Velcescu, Aleandra Alecu, Gabriela Florea, Programare avansată in ORACLE9i, Editura Tehnica, Bucureşti, 2004
3. Felicia Ionescu, Baze de date Realationale si aplicatii, Editura Tehnica, Bucuresti, 2004
4. Ion Lungu, ManoleVelicanu, Bodea C, Ionită C, Sisteme de gestiune a bazelor de date (SGBD) si aplicatii Oracle, Ed.All, Bucuresti ,1998
5. Marin Fotache, Proiectarea bazelor de date, Ed.Polirom,Iaşi, 2005
6. Ion Lungu, Manole Velicanu,Badescu G, Ionita C,Sisteme de Gestiune A Bazelor de date, Ed.Petrion,Bucuresti, 2000
7. Marin Fotache, Cretu Liviu, Catalin Strimbei, Oracle 9i –ghidul dezvoltarii aplicatiilor profesionale,Ed,Polirom , Iasi, 2003
8. Marin Fotache, Cretu Liviu, Catalin Strimbei,Ioan Brava, Visual Foxpro –ghidul dezvoltarii aplicatiilor profesionale,Ed,Polirom , Iasi,2001
9. Marin Fotache, SQL Dialecte DB2, ORACLE, VISUAL FOXPRO, Ed.Polirom Iasi,2001
10. Tom Luers, Bazele Oracle 7, Ed.Teora , 1996
11. Popescu Ileana, Modelarea bazelor de date, Ed.Tehnica, Bucuresti,2001
12. ***,Oracle 9i Aplication Developer's Guide-Fundamentals,Oracle Corporation,2002
13. ***,Oracle 9i Database Concepts,Oracle Co.,2002
14. ***,Oracle 9i –Documentatia ,Oracle Co.2001
15. Popescu Ileana, Bazele de date relationale, Ed.Universitatii din Bucuresti,1996
16. Connolly, T.,Begg,C.,Baze de date-Proiectare,Implementare,Gestionare ,Ed.Teora, 2001
17. Dollinger ,R,Baze de date si Gestiunea Tranzactiilor,Ed.Albatros, Cluj-Napoca,1997
18. Carmen Petre, Daniela Popa,Iliescu C,Metodica predarii informaticii si tehnologiei informatiei, Ed.Arves,2002
19. M.E.N si C.N.C ,Gihid Metodologic pentru Informatica si Tehnologia informatiei,Ed.Aramis,Bucuresti,2001
20. S.N.E.E ,Ghid de evaluare informatica si tehnologia informatiei,Ed.Aramis,Bucuresti,2001
21. Ioan Jinga, Elena Istrate, Manual de pedagogie, Ed.All, 1998
22. Julie C.Meloni,Php,MySQL si Apache,Ed.Corint, 2005
23. Adrian Munteanu, Valerica Serban, Gabriel Cristescu, Retele Windows (2003), servere si clienti.exemple practice, Ed.Polirom , Iasi, 2004
24. Adrian Munteanu,Valrica Serban, Retele locale de calculatoare, proiectare si administrare, Ed.Polirom,Iasi, 2003
25. Sabin Buraga, Aplicaţii WEB, implementări în php, Ed.Polirom Iaşi, 2003